

APPUNTI DELLE LEZIONI DEL CORSO DI DECISION MODEL 2020/2021

Alessandro Maccario

DISCO - Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi Milano-Bicocca
Italy
2020/2021

Contents

1 DECISION MODEL	5
2 SECONDA LEZIONE - 10-03-2021	7
3 TERZA LEZIONE	13
3.0.1 Cos'è l'ottimizzazione matematica?	13
3.0.2 I 5 passi per la formulazione del modello di programmazione lineare	16
4 QUARTA LEZIONE	23
4.1 Risolvere un problema di programmazione lineare: un approccio grafico	26
5 QUINTA LEZIONE 17-03-2021	31
5.0.1 Sensitivity Analysis	34
6 SESTA LEZIONE - Practical Session 1	35
6.0.1 Finding the optimal solution graphically	36
6.0.2 Finding the optimal solution analytically	39
6.0.3 Standard form	39
6.0.4 Basic variables	40
7 SETTIMA LEZIONE - Practical Session 2	43
7.0.1 Dissection of the Problem	44
7.0.2 Dissection of the problem	46
8 LEZIONE 6 - L'algoritmo del Simplex	47
8.1 Sensitivity Analysis	50
9 LEZIONE 7 - Sensitivity ex. 6-7	57
9.0.1 Primo scenario	58
9.0.2 Secondo scenario	60
10 LEZIONE 8 - INTEGER PROGRAMMING	65
10.0.1 Algoritmo di Branch-and-Bound	70
11 LEZIONE 9 - Practical Session - Integer Programming (14/04/2021)	77
12 LEZIONE 10 - (19/04/2021)	91
13 LEZIONE 11 - Practical Session Ex. 10 - Network Modelling (21/04/2021)	103
14 LEZIONE 12 - Webex Lecture n. 9 (26/04/2021) - Non linear Programming Problems	113
14.0.1 Non Linear Programming: Introduction	126

Chapter 1

DECISION MODEL

This course is based main in **prescriptive** analytics (while **descriptive** and **predictive** analytics has become well-established).

Prescriptive: answer to the question "What should I do?".

The final objectives is: transforming data in good prediction and therefore in good decisions.

What is **analytics**? Different definitions:

- Analytics is the **discovery, interpretation, and communication of meaningful patterns in data**. Especially valuable in area rich with recorded information, analytics relies on the simultaneous application of **statistics, computer programming and operations research** to quantify performance. (Wikipedia definition)
- The field of data analysis. Analytics often involves studying **past** historical data to research potential **trends**, to analyze the **effects** of certain decisions or events, or to evaluate the performance of a given tool or scenario. The goal of analytics is to improve the business by **gaining knowledge** which can be used to make **improvements** or changes. (business dictionary - analytics definition)

In our course we refer to analytics as:

- Using data to build models that lead to better decisions;
- Ultimately create value;
- Can use **big data** or **small data**;

We have different type of analytics:

- **Descriptive analytics**: we find in it **descriptive statistics** such as **sampling, mean, mode, median, standard deviation, range and variance, steam and leaf diagram, histogram, interquartile range, quartiles, frequency distributions**;
- **Predictive analytics**:
 - **Forecasting**: such as **Time Series, Casual relationships**;
 - **Data Mining**: such as **Cluster analysis, Association Analysis, Multiple Regression, Logistic Regression, Decision Tree Methods, Neural Networks, Text Mining**;
- **Prescriptive analytics**:
 - **Management Science**: such as **linear programming, sensitivity analysis, integer programming, goal programming, nonlinear programming, transportation, logistics, optimization heuristics, simulation modeling**

To resume all what we've been said:

- **Descriptive**: finds patterns in the data, such as:

- Summary Statistics;
- Visualizations;
- Clustering;
- etc.

- **Predictive:** predict different outcomes, such as:

- Linear regression;
- Logistic regression;
- CART;
- Random Forests;
- ML;
- etc.

- **Prescriptive:** gives advice on actions to take, such as:

- Optimization;

What is Analytics?

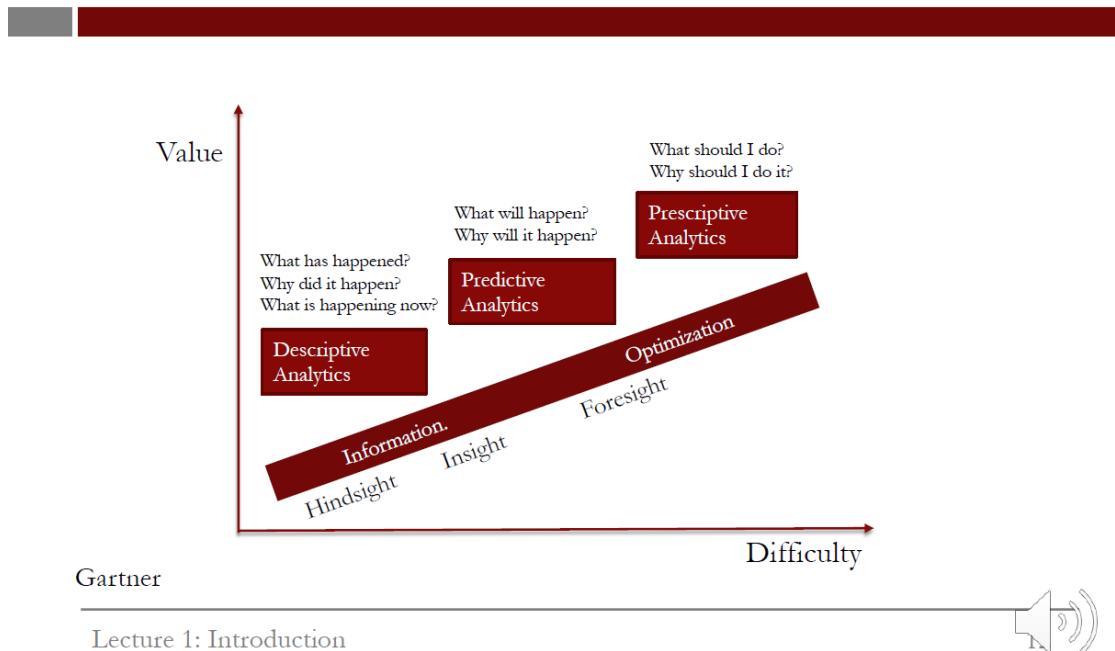


Figure 1.1: Analytics

Chapter 2

SECONDA LEZIONE - 10-03-2021

- **Descriptive Analytics:** descrive le caratteristiche dei dati principali a disposizione (ne determina le statistiche, si visualizzano e si perlustrano i dati). L'obiettivo è di fare un report che mi descrive i dati a disposizione.
- **Predictive Analytics:** partendo dalla *descriptive* (quindi dall'analisi dei dati precedenti), si estraggono correlazioni, modelli di interdipendenza tra variabili che ci permettono di prevederne l'andamento: esempio classico della *regressione*; oppure, se sono una compagnia assicurativa posso analizzare i sinistri accaduti in passato, analizzarli, e prevedere il numero di sinistri che potrebbero accadere in futuro o capire se alcune denunce possono essere o meno fraudolente. Comincio a capire quindi come i dati sono correlati tra loro, le interdipendenze tra i dati e creo modelli che spiegano ciò che è successo che mi aiutano a prevedere quello che potrebbero succedere. Quali sono i tools? Sono tutti i tutti inerenti la regressione e tutti i modelli di *Machine Learning*; (tirare conclusioni e prevedere futuro)
- **Prescriptive Analytics:** parte dalla *predictive analytics* e contestualizza queste previsioni nel sistema che sto studiando. Comincia a studiare quali sono le implicazioni delle previsioni e anche a capire come in vista delle previsioni o dei dati di quello che penso succederà, io posso agire sul sistema per raggiungere degli obiettivi. Con la prescriptive, per la prima volta, entrano in gioco gli **obiettivi** della mia ricerca e devo trovare il modo per raggiungerli. Ad esempio: posso trovare il miglior insieme dei prezzi e la frequenza con cui faccio pubblicità per massimizzare i miei ritorni in una campagna di marketing; o trovare le mosse giuste di business per trovare un giusto obiettivo; o per trovare un piano di distribuzione ottimale per la spedizione di pacchi. Nella *prescriptive analytics* entrano in gioco le decisioni. Quindi significa prendere delle decisioni basate sui dati. Alcune tecniche che possiamo considerare: **linear programming, sensitivity analysis, integer programming, goal programming, non linear programming, simulation modeling**. Alcuni esempi:
 - **Industry 4.0:** industria della quarta rivoluzione industriale, che usa l'automazione per rendere più efficiente tutti i processi di produzione e supply chain. Industria che produce, scambia e utilizza i dati per creare processi che rendano più efficiente la produzione e le condizioni di lavoro. Prendiamo l'esempio della *manutenzione predittiva*: ci sono sensori collegati alle varie macchine che generano i dati e che possono dare indicazioni sul funzionamento delle macchine stesse (stessa cosa usata nel calcio). L'obiettivo della manutenzione predittiva è quello di non aspettare che la macchina si rompa, ma di pianificare una serie di interventi di manutenzioni che possano mantenere attivo ed efficiente il processo di produzione, ovviamente riducendo anche i costi della manutenzione predittiva (non posso controllare tutte le macchine). Ho quindi bisogno della *descriptive analytics* che mi facciano vedere qual è la situazione corrente del mio sistema di produzione per poi creare dei *modelli predittivi* che prevedano che una certa situazione potrebbe generare un guasto. E quindi mi dà delle probabilità di guasto delle macchine nel mio sistema di produzione. Devo quindi programmare i miei interventi, ovvero quali interventi mettere in atto minimizzando i costi necessari per farlo. Per la *predictive analytics* quindi ho bisogno di tutti e tre gli strumenti di analytics.

- **Trasporti:** oltre alle indicazioni stradali (Esempio: Google Maps) contengono anche algoritmi predittivi che si basano sui dati passati per fare previsione del traffico e quindi specificare il tempo di percorrenza fra il punto A e il punto B.

Quindi noi ci mettiamo nella *Prescriptive analytics* riguardante i modelli decisionali.

Cos'è il **Decision making?** Processo di selezione di una soluzione logica. In generale, si tratta di scegliere una decisione tra tante diverse opzioni basandosi anche sui dati a disposizione. Quando prendo una decisione devo considerare tutte le alternative: il concetto è di *pesare gli aspetti positivi e negativi delle opzioni*. Considero i pro e i contro, li soppeso e scelgo quale sembra il migliore.

Nel business, per fare delle scelte che siano efficaci e che portino a dei buoni risultati, la persona deve essere in grado di prevedere le conseguenze di questa scelta e quindi devo avere un modello che mi fa capire quali sono le implicazioni della scelta che io posso fare, qual è la relazione tra le scelte che faccio e l'outcome che potrei ottenere. Quindi l'outcome è sempre relativo ad un obiettivo (massimizzare le prestazioni di un sistema, minimizzare i costi etc.). Quindi, quando prendo una decisione ho sempre in mente un **obiettivo** e le mie decisioni devono essere tali affinché possa raggiungere al meglio il mio obiettivo. Per prendere decisioni informate uso la **descriptive** e **predictive analytics**.

Quali tipi di decisioni possono esserci? In che cosa si differenziano le decisioni? Esse hanno caratteristiche diverse e in base alle loro caratteristiche devo utilizzare determinati modelli per arrivare ad una soluzione ottimale:

- **Unstructured problems:** (quando parliamo di struttura parliamo di struttura matematica) problemi che sono nuovi e per i quali le informazioni sono ambigue o incomplete (non legati ad una routine); sono problemi che richiedono soluzioni custom-made (su misura). Sono problemi che devo affrontare in condizioni di incertezza e che richiedono una soluzione customizzata. Riguardano decisioni *non programmate, uniche e non ricorrenti*: a quale corso di studio mi iscrivo? Ho alcune informazioni incomplete e ambigue e quindi molta incertezza su come saranno i diversi corsi di studio e questa è un *problema non strutturato* in quanto c'è molta incertezza su come saranno i diversi corsi di studio. Le decisioni che vengono prese sono definite **decisioni strategiche** e che influenzano un orizzonte temporale abbastanza lungo venendo prese una volta ogni tanto; (Non-programmed decision)
- **Structured problems:** in questo caso gli obiettivi sono chiari, familiari (sono già capitati in passato quindi conosco bene il problema), sono facili e completamente definiti (routine approach). Si tratta di applicare un metodo per risolvere il problema: decisione ripetitiva che viene trattata come un approccio di routine. Esempio: Amazon avrà dovuto decidere dove costruire i propri warehouse (magazzini) e quindi quale fosse la capacità dei magazzini, della sua flotta, o che tipo di contratto fare verso i fornitori etc. Queste sono decisioni strategiche che non vengono fatte tutti i giorni e per le quali trovare un modello diventa più difficile. Una volta che arrivano le domande di acquisto, i problemi di allocazione ai corrieri (o di routing: quale strada prendono i corrieri per essere più veloci?) delle spedizioni etc. è un problema che può essere un problema di routine. Data le consegne che devo fare determino il percorso ottimale che i diversi veicoli devono fare. Problema che posso modellare in termini matematici e risolvere routinariamente. Seppur ci sia incertezza sul traffico e tante altre cose, studio un modello nel quale ogni volta inserisco dati diversi ma non vado a modificare il modello. Problema più strutturato e modellizzabile tramite un modello matematico.

Quali sono le implicazioni di questi problemi a livello manageriale?

1. **Problemi strutturali** (operazionali): sono tipicamente problemi di ottimizzazione a *breve termine* (orizzonte di decisione a breve termine);
2. **Problemi tattici/manageriali;**
3. **Problemi non strutturali:** tipicamente sono problemi strategici, il cui orizzonte temporale sulla quale la mia decisione avrà effetto, sarà un orizzonte temporale più lungo, ad esempio di anni.

Sono decisioni *strutturali*, che riguardano la struttura del mio problema e che sono decisioni più strategiche;

Quindi, che cosa caratterizza le **decisioni operative**? In genere quando ho un problema operativo, il numero di possibili soluzioni è *altissimo*. Ad esempio per il problema della distribuzione, abbiamo le richieste di consegna di Amazon e dobbiamo decidere quale percorso far fare ai corrieri: si hanno tantissime soluzioni! Quanto ci mettiamo a valutare queste soluzioni? Tendenzialmente molto poco: una volta data la stima di per correnza di un certo tratto di strada, capire quanto tempo ci vorrà per fare determinate consegne è una funzione facile da calcolare/approssimare. Quindi nei *problemi strutturati* (operativi) ho tantissime soluzioni da scegliere. Anche se ogni singola soluzione è valutabile in tempi brevi, il fatto che siano tantissime mi crea la difficoltà del problema. Devo trovare tra tutte le possibili soluzioni di routing, quella ottimale. Lo stesso vale per l'orario delle lezioni (è un problema strutturato) che può essere modellato in termini matematici (è un problema combinatorico) e trovare la soluzione ottima che potrebbe richiedere un tempo lungo (ci possono essere tante soluzioni da valutare). Nelle soluzioni di *problemi strategici* (quindi non strutturati), le soluzioni sono pochissime: nel decidere il percorso di studi avevamo due o tre soluzioni. Lo stesso quando bisogna decidere dove costruire uno stabilimento o un magazzino ad esempio: le possibili scelte sono pochissime. La difficoltà è: una volta decisa una soluzione è difficile capire quale sia l'outcome della soluzione anche perché aumenta l'orizzonte temporale della mia decisione. Aumenta l'incertezza su quali siano le ripercussioni delle mie decisioni.

In questa introduzione parliamo di ottimizzazione in tema di Business. In realtà, oltre al business, l'ottimizzazione ha anche un'altra connessione col Machine Learning, ovvero l'ottimizzazione utilizzata per l'apprendimento del modello, quindi un altro ruolo dell'ottimizzazione. L'ottimizzazione utilizzata per apprendere un modello di ML deve trovare il valore ottimo dei parametri del modello (è decisione operativa: se dobbiamo trovare i parametri ottimali di una rete neurale, abbiamo i dati di traininini, addestriamo la rete neurale e un algoritmo di ottimizzazione che è automatizzato, prenderà le decisioni ottime nel senso che troverà i parametri che ottimizzano le prestazioni di quella rete neurale. Sicuramente non è un problema strategico. Quindi l'ottimizzazione applicata al ML risolve problemi *operativi*).

Dal punto di vista manageriale abbiamo una complessità crescente passando da **problemi strutturati** (una volta capito e codificato, diventa una cosa automatica) a **problemi non strutturali** (il problema della valutazione dei pro e dei contro di una certa soluzione è un problema complesso, che richiede raccolta di dati, modelli previsionali e una corretta analisi di tutte queste informazioni e che ha un alto livello di incertezza. Oltre tutto i problemi non strutturati sono problemi che non mi si presentano tutti i giorni e come tutti i problemi nuovi sono più complessi). A livello manageriale si dice che la complessità va da problemi strutturati a problemi non strutturati e diventa sempre più elevata (la complessità cresce con l'orizzonte temporale della decisione e il tipo di decisione). Quando però si parla di *complessità* essa ha significati molto diversi:

- si parla di complessità di un algoritmo ad esempio, per valutare tutte le possibili soluzioni in un problema operativo;
- dal punto di vista dell'ottimizzazione, complessità significa, a livello di ottimizzazione combinatoria che il numero delle possibili soluzioni da valutare può essere un *indice di complessità* (posso avere il *problema del commesso viaggiatore* (Traveling salesman problem - TSP) che deve visitare tutti i nodi di un grafo senza tornare mai nello stesso nodo);
- la complessità di un problema possono essere i diversi criteri, l'obiettivo può comporsi di diversi pezzi: tempo/costo, rischio/rendimento. Posso avere obiettivi diversi anche in contrasto tra loro: massimizzare il rendimento di portafoglio in genere massimizza anche il rischio. Quindi trovare un trade-off in questi casi, bisogna trovare un metodo per trovare un compromesso tra questi obiettivi. Più il numero di criteri è elevato e più il problema può essere complesso;
- altra fonte di complessità è l'**incertezza**: se conosco i dati del mio problema, lo risolvo e se trovo una soluzione so che è quella ottimale. Se devo risolvere un problema in condizioni di **incertezza**, come definisco la soluzione ottimale? Supponendo diversi scenari e supponendo di poter risolvere

il problema in tutti questi scenari (quindi avendo diverse soluzioni), la domanda in un contesto incerto è: qual è la soluzione ottima fra tutte queste? Il concetto di ottimalità in un contesto incerto è difficile da definire! (potrei usare il valore atteso dei diversi scenari, ma non necessariamente è la soluzione migliore). La scelta si complica quando si agisce in condizione di incertezza.

Quindi:

- Quando agisco in condizioni di **Certezza**: tutte le possibili conseguenze delle mie decisioni sono conosciute e posso determinare se la mia soluzione è ottima oppure no;
 - Quando agisco in condizioni di **Incertezza**: non conosco le possibili conseguenze delle mie decisioni. Se ho delle probabilità posso determinare la probabilità di *likelihood* delle mie scelte, costruendone una distribuzione di probabilità (quindi un valor medio, una varianza che mi definiranno un indice di rischio, ma il concetto di decisioni ottima è più difficile da definire).
-

Sui modelli non strutturati staremo ben poco (non è un corso di management). Quando posso usare un modello questo mi aiuta a prendere delle decisioni anche se penso di prendere queste decisioni semplicemente basandomi sull'intuito. Vediamo quindi alcuni *bias* che si possono avere basandoci solo sull'intuito. E' quindi importante quando si prendono delle decisioni o quando dobbiamo giustificare una decisione che prendiamo anche a livello manageriale, usare dei modelli che ci garantiscano la qualità della decisione. Vediamo due tipi di **effetti** che possono esistere quando si prendono decisioni basate sull'intuito: (si possono avere dei **bias cognitivi**, dei modi di pensare che possono influenzare le mie decisioni)

- **Anchoring Effects:** il decisore si ancora ad una informazione che ha che gli impedisce di avere una visione completa del problema. Ad esempio: è stato chiesto ad un gruppo significativo di persone di dire instintivamente quanto facesse

$$1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8$$

La mediana delle risposte fu di 512. Allo stesso gruppo di persone fu chiesto poi di dire quanto facesse, invece, il seguente calcolo:

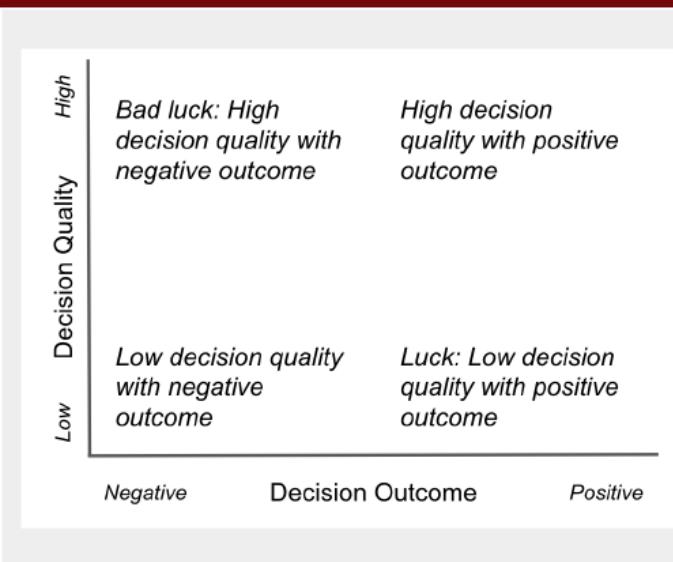
$$8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$$

La mediana della risposta durante l'esperimento fu di 2250. La risposta fu **bias**. Viene naturale, se si parte da numeri piccoli, pensare ad un risultato minore. Mentre se la domanda partisse da numeri più elevati, ci si **ancora** a risultati più elevati. Questa domanda non chiede altro di dire quanto fa $8!$, ovvero 40320. Ci si ancora ad un bias mentale che influenza la risposta, quindi ad una visione parziale del problema. Si trovano quando elementi triviali influenzano il pensiero rispetto al problema. Il pensiero si ancora ad una visione parziale del problema. Quando esprimiamo il nostro giudizio rispetto ad un problema, ci possono essere questi aspetti di anchoring, framing che influenzano le nostre decisioni che ci impediscono di prendere decisioni razionali (non c'è nessuna ragione per cui i due calcoli precedenti siano diversi!);

- **Framing effects:** ci sono anche gli effetti di framing (framing concettuale: c'è un elemento che mi impedisce di affrontare razionalmente il problema). Se pensiamo ad un quadro al quale viene applicata una cornice rossa, gli elementi rossi di quel quadro verranno esaltati. Se viene usata una cornice blu si esalteranno gli elementi blu all'interno di quel quadro. Ho quindi un frame concettuale che mi impedisce di vedere bene e in modo razionale il problema. Anche facendo aggiustamenti rischio di non decidere in modo razionale.

Quindi, ci sono buone decisioni e buoni risultati che non necessariamente coincidono. I modelli di decisione ci aiutano a prendere buone decisioni ma non garantiscono il risultato ottimale (sarebbe molto avere dei modelli che ci garantiscono il risultato ottimale). Quando si agisce in condizioni di incertezza porta a tali problemi). Nonostante ciò, usare un approccio di modellazione che cerchi di dare una struttura al problema, può sicuramente generare dei buoni risultati più frequentemente rispetto a prendere decisioni in modo azzardato.

Good Decisions vs. Good Outcomes



<https://app.wooclap.com/events/URDZEQ/questions/6047e628790a4311025c0360>

9

Figure 2.1: Good Decisions versus Good outcomes

Chapter 3

TERZA LEZIONE

Oggi iniziamo a focalizzarci su dei modelli di ottimizzazione tipici della scienza delle decisioni e ne vedremo le parti fondamentali (non approfondiremo tutti gli aspetti tecnici/teorici dei modelli, ma come si possono utilizzare e a cosa servono). Vediamo il **problema dell'ottimizzazione**.

3.0.1 Cos'è l'ottimizzazione matematica?

Ottimizzazione viene dalla stessa radice di *ottimale*, il che significa il meglio. Quando si ottimizza qualcosa, lo si sta renderlo *il migliore*. Ma il *meglio* può cambiare: se si è un calciatore, si vogliono massimizzare i metri corsi o minimizzare le perdite di palla; se si è un investitore può voler significare ottimizzare il rendimento e minimizzare i rischi.

L'ottimizzazione matematica è quella branca della matematica che ci aiuta a prendere delle decisioni sia *massimizzando* che *minimizzando* un certo obiettivo. Il nostro focus sarà nella modellazione e valutazione delle soluzioni piuttosto che nel metodo risolutivo. Le applicazioni che vediamo dell'ottimizzazione matematica le vediamo tutti i giorni: i magazzini di Amazon hanno bisogno di ottimizzazione per la gestione del magazzino; le compagnie aeree hanno bisogno di ottimizzazione in termini di allocazione o la determinazione dei prezzi degli aerei; il problema degli orari è un problema di ottimizzazione (lo scheduling, i turni); il problema della distribuzione dell'energia; problemi finanziari, problemi di marketing. In quasi tutti i settori della vita reale ci sono problemi di ottimizzazione.

Con l'ottimizzazione matematica si trasforma in una funzione matematica il problema e poi abbiamo bisogno di algoritmi che trovino il massimo o il minimo di queste funzioni matematiche. Cos'è la cosa più importante in questo caso? Il saper formulare bene il problema o il saperlo risolvere bene? Se io formulo male il problema, posso avere l'algoritmo migliore del mondo ma non avere la soluzione corretta. In generale la programmazione matematica può tenere conto anche di limiti in termini di risorse per lo spazio decisionale entro il quale deve essere trovata una soluzione ottima.

Limitare lo spazio decisionale può facilitare o complicare il problema? Dal punto di vista applicativo, limita il numero di soluzioni, ma rende computazionalmente più complicato raggiungerla. Noi ci troviamo nell'ambito di avere una **funzione obiettivo** e abbiamo dei vincoli espressi sotto forma di funzioni matematiche e useremo degli algoritmi per trovare la soluzione ottima. Questo fa sì che risolvere il problema di ottimizzazione tenendo conto dei vincoli, limita le possibili soluzioni ma aumenta la richiesta computazionale, rende più difficile risolvere il problema. Dal punto di vista matematico, inserire dei vincoli complica le cose dal punto di vista computazionale. Perché? Se abbiamo una funzione di qualsiasi tipo e dobbiamo massimizzarla o minimizzarla cosa facciamo? Troviamo i massimi e i minimi di queste funzioni. Come? Dalle derivate uguali a 0 e poi valutiamo i punti per vedere se sono punti di massimo o di minimo. Al di là dal fatto che possiamo avere tanti punti che possono avere derivata uguale a 0 e che potrebbero essere o massimi o minimi e quindi potremmo avere un problema di ottimizzazione globale rispetto ad una ottimizzazione locale, se poi però ci mettiamo anche delle funzioni che limitano il dominio della nostra funzione obiettivo, ci complichiamo la vita perché dobbiamo verificare che la nostra soluzione sia dentro la regione oppure no. Lo vedremo più nel dettaglio avanti. In generale, **aggiungere**

dei vincoli può complicare la parte computazionale. Questo nella programmazione non lineare, mentre nella **programmazione lineare** accade che aggiungere dei vincoli semplifichi la soluzione.

Quindi la **programmazione matematica** (detta anche *ottimizzazione matematica*) sono le tecniche matematiche che si occupano di trovare l'ottima di una funzione considerando eventualmente anche dei vincoli. Si usa il termine *programmazione* perché nasce in un ambito in cui io dovevo *programmare* delle attività, fare un piano delle attività e quindi dovevo programmare delle scelte, delle decisioni.

Quali sono le caratteristiche principali di un problema di ottimizzazione?

- **Variabili di decisione:** ci sono delle decisioni da prendere. Come le prendo? Andando a massimizzare/minimizzare un certo obiettivo, ovvero una certa *funzione obiettivo*;
- **Funzione obiettivo:** devo avere un obiettivo da massimizzare/minimizzare (che è funzione delle variabili di decisione cercando di soddisfare dei *vincoli*);
- **Funzioni di vincolo:** vincoli che queste variabili dovranno soddisfare;

Quindi la funzione obiettivo rappresenta l'obiettivo del problema che può essere sia massimizzata sia minimizzata. Le variabili sono le variabili sconosciute di cui vogliamo trovare il valore. Quindi bisogna distinguere nella programmazione matematica le *variabili* dai *parametri*: le **variabili** sono quelle leve su cui possiamo agire per cercare di ottimizzare la *funzione obiettivo*, quindi sono sconosciute e devo determinare il valore ottimo e i *parametri* sono i *dati* del modello. Infine, abbiamo una **serie di vincoli** che limita il valore di queste variabili in un certo spazio. Quindi il problema di ottimizzazione alla fine è quello di trovare il valore delle variabili che minimizzano o massimizzano la funzione obiettivo soddisfando i vincoli.

La forma generale di un problema di ottimizzazione:

$$\begin{aligned}
 & \text{MAX (or MIN)} \quad f_0(X_1, X_2, \dots, X_n) \\
 & \text{Subject to :} \quad f_1(X_1, X_2, \dots, X_n) \leq b_1 \\
 & \quad f_k(X_1, X_2, \dots, X_n) \geq b_k \\
 & \quad f_m(X_1, X_2, \dots, X_n) = b_m
 \end{aligned} \tag{3.1}$$

Quindi ho delle *variabili di decisione* che sono un vettore/matrice di variabili di decisione in uno spazio \mathbb{R}^n ; abbiamo una *funzione obiettivo* definita su queste variabili di decisione e ho dei *vincoli* che possono in teoria essere di \leq o di \geq . Nella programmazione lineare tutti questi vincoli saranno lineari; nella programmazione non lineare potrò anche avere delle funzioni non lineari. Potrò poi avere dei problemi dove addirittura posso non conoscere esattamente la funzione obiettivo, perché in modalità più complesse, la funzione obiettivo potrebbe essere il risultato di una simulazione che è difficilmente esprimibile in forma analitica. In questo contesto ci limiteremo a considerare casi in cui è possibile avere una formulazione analitica di queste funzioni, quindi conosciamo l'espressione analitica di queste funzioni. I vincoli che vediamo con minore/maggiore uguale è possibile renderli come vincoli di maggiore/minore, basta cambiare segno, moltiplicando a destra e sinistra per -1 capovolgendo la disegualanza. Ma è anche possibile rendere un vincolo di uguaglianza come due vincoli, uno di minore e l'altro di maggiore.

Vediamo un problema di clustering: il problema è che dati m punti x_1, \dots, x_m nello spazio $n - \text{dimensionale } \mathbb{R}^n$, e un numero k di cluster, devo determinare k centroidi tali che la somma delle distanze da ogni punto al cluster più vicino sia minimizzata. Ciò che devo fare è trovare questi k centroidi c_1, \dots, c_k , che minimizzino la sommatoria da $1, \dots, n$, ovvero:

$$\min_{c_1, \dots, c_k} \sum_{i=1}^m \min_{l=1, \dots, k} \|x_i - c_l\| \tag{3.2}$$

$$\begin{aligned}
 & \underbrace{\text{minimize}}_{c_l, t_{il}} \sum_{i=1}^m \sum_{l=1}^k t_{il} \cdot \|x_i - c_l\| \\
 & \text{subject to } \sum_{l=1}^k t_{il} = 1, \quad t_{il} \geq 0, \quad i = 1, \dots, m, \quad l = 1, \dots, k
 \end{aligned} \tag{3.3}$$

dove c è il centroide (anch'esso deve essere minimizzato in quanto non si conosce all'inizio). Ma come posso risolvere tale problema? In questa formula le x sono i punti non ci sono le variabili decisionali, quindi x è solo un dato del problema, ma io le x le conosco! Quello che voglio fare è assegnarli in modo ottimale ai centroidi e determinare quali sono questi centroidi. Ho bisogno quindi delle variabili di decisione. Quali sono le decisioni che devo prendere? Sono decisioni di *assegnamento*. Devo assegnare i miei punti. Questa variabile è quella che è stata definita come t_{il} che varrà 1 se ho associo il punto i al cluster l , altrimenti varranno 0. A questo punto se introduco queste variabili devo introdurre dei vincoli, altrimenti non determino il valore delle variabili t_{il} . Queste variabili mi aiutano perché mi dicono che un punto può appartenere solo ad un cluster, quindi la sommatoria che va da $1, \dots, k$ per t_{il} deve essere uguale a 1 e t_{il} deve essere ≥ 0 . Quindi, questo vincolo mi dice che devo associare un punto ad un cluster e ad uno solo, e quindi il modello è forzato a fare un assegnamento.

Riprendiamo quindi dalla formulazione generale del problema di ottimizzazione e inizieremo oggi a vedere la **programmazione lineare** che è la parte più semplice da cui partire per andare a capire i concetti principali dell'ottimizzazione. Nella programmazione lineare noi abbiamo le funzioni *obiettivo* e le funzioni dei *vincoli* che sono tutte funzioni **lineari**, che possono essere rappresentate in questa forma:

$$\begin{aligned}
 & \text{MAX (or MIN)} \quad c_1X_1 + c_2X_2 + \cdots + c_nX_n \\
 & \text{Subject to :} \quad a_{11}X_1 + a_{12}X_2 + \cdots + a_{1n}X_n \leq b_1 \\
 & \quad a_{k1}X_1 + a_{k2}X_2 + \cdots + a_{kn}X_n \geq b_k \\
 & \quad a_{m1}X_1 + a_{m2}X_2 + \cdots + a_{mn}X_n = b_m
 \end{aligned} \tag{3.4}$$

Esempio:

An Example LP Problem

Blue Ridge Hot Tubs produces two types of hot tubs: Aqua-Spas & Hydro-Luxes.

	Aqua-Spa	Hydro-Lux
Pumps	1	1
Labor	9 hours	6 hours
Tubing	12 feet	16 feet
Unit Profit	\$350	\$300

There are 200 pumps, 1566 hours of labor, and 2880 feet of tubing available.

Figure 3.1: Linear Programming Problem Example

Ricordare: nei problemi di programmazione lineare i vincoli sono *sempre* \leq , \geq , oppure $=$. Perché? Perché la soluzione si basa su un sistema di equazioni e quindi ci deve essere l'uguale. Ciò non significa

che non sia possibile rappresentare dei vincoli di stretta minoranza o di stretta maggioranza, bisognerà aggiungere in quel caso un'altra variabile di decisione definita come ϵ che sarà da minimizzare o piccolo a piacere.

3.0.2 I 5 passi per la formulazione del modello di programmazione lineare

1. Understand the problem;
2. Identificare le variabili di decisione (quali sono le cose che voglio decidere e come le formalizzo?). In questo esempio specifico abbiamo che:

X_1 = numero di Acqua-Spas da produrre; X_2 = numero di Hydro-Luxes da produrre;

3. Stabilite le variabili di decisione, formuliamo la funzione obiettivo come funzione lineare. In questo caso la mia funzione obiettivo è quella di *massimizzare il profitto* e quindi avrò che:

$$\text{MAX: } 350X_1 + 300X_2$$

4. Stabilita la funzione obiettivo si deve formulare i vincoli:

$$1X_1 + 1X_2 \leq 200 \quad \text{Pumps} \quad 9X_1 + 6X_2 \leq 1566 \quad \text{Labor} \quad 12X_1 + 16X_2 \leq 2880 \quad \text{Tubing}$$

determinando i 3 vincoli che corrispondono alle tre risorse scarse che mi limita dal produrre infinite vasche.

5. Identificare gli *upper-bound* e i *lower-bound* sulle variabili di decisione. Molto spesso quello che abbiamo è il *lower-bound* di non negatività, ovvero la quantità prodotto per ogni tipologia di vasca idromassaggio dovrà essere maggiore di 0. Questo è un vincolo che aiuta la risoluzione del problema: a differenza di altri vincoli che possono rendere computazionalmente pesante la risoluzione del problema, i vincoli di non negatività rendono più facile la sua risoluzione. Se non ci fosse dovremmo ricavarcelo per il software. Quindi:

$$X_1 \geq 0 \quad X_2 \geq 0$$

A questo punto abbiamo modellato il problema e la modellazione del precedente esempio può quindi essere così riscritta come:

$$\begin{aligned} & \text{MAX : } 350X_1 + 300X_2 \\ & \text{S.t. : } 1X_1 + 1X_2 \leq 200 \\ & \quad 9X_1 + 6X_2 \leq 1566 \\ & \quad 12X_1 + 16X_2 \leq 2880 \\ & \quad X_1 \geq 0; \quad X_2 \geq 0; \end{aligned} \tag{3.5}$$

Quindi abbiamo detto: programmazione lineare, quindi funzione obiettivo e vincoli sono lineari, i cui vincoli possono essere di maggiore/minore/uguale. Dal problema precedente vediamo che la prima vasca prodotta ha il profitto maggiore! L'idea potrebbe essere quella di produrre più X_1 possibili. Ovvero:

Solving LP Problems: An Intuitive Approach

MAX: $350X_1 + 300X_2$
S.T.: $1X_1 + 1X_2 \leq 200$
$9X_1 + 6X_2 \leq 1566$
$12X_1 + 16X_2 \leq 2880$
$X_1 \geq 0$
$X_2 \geq 0$

- Idea: Each Aqua-Spa (X_1) generates the highest unit profit (\$350), so let's make as many of them as possible!
- How many would that be?
 - Let $X_2 = 0$
 - 1st constraint: $1X_1 \leq 200$
 - 2nd constraint: $9X_1 \leq 1566$ or $X_1 \leq 174$
 - 3rd constraint: $12X_1 \leq 2880$ or $X_1 \leq 240$
 - If $X_2=0$, the maximum value of X_1 is 174 and the total profit is $\$350*174 + \$300*0 = \$60,900$
 - This solution is *feasible*, but is it *optimal*?

Figure 3.2: Linear Programming Problem Solution

La soluzione è sicuramente **ammissibile** (*feasible*: soddisfa tutti e tre i vincoli), ma siamo sicuri sia la soluzione *ottimale*? Rimangono delle risorse inutilizzate e quindi posso chiedermi se magari posso combinare in modo migliore le mie risorse per avere una combinazione che riesce ad utilizzare meglio le risorse che ho a disposizione. In ogni caso, questa prima soluzione è una **soluzione ammissibile** (una soluzione **ammissibile** soddisfa tutti i vincoli del problema). E' quindi una decisione implementabile e possibile). Vediamo se riusciamo a trovare una soluzione migliore.

- I **vincoli** di un problema di programmazione lineare definiscono la sua **regione ammissibile** e il punto della regione ammissibile con la soluzione migliore è appunto la **soluzione del problema**;
- per i problemi con due variabili (caso molto ridotto) è facile disegnare la regione ammissibile. La soluzione grafica è un metodo per spiegare il funzionamento generale dell'algoritmo di risoluzione, perché fa vedere geometricamente cosa succede, ci aiuta a comprendere il funzionamento di questi modelli di programmazione lineare e di cosa vuol dire trovare il punto di ottimo di questi modelli, ma sicuramente non perché è un metodo largamente utilizzato (i problemi a due variabili nella realtà sono pochissimi);

Con i vincoli di non negatività la regione ammissibile è il quadrante con $X_1, X_2 \geq 0$, ovvero:

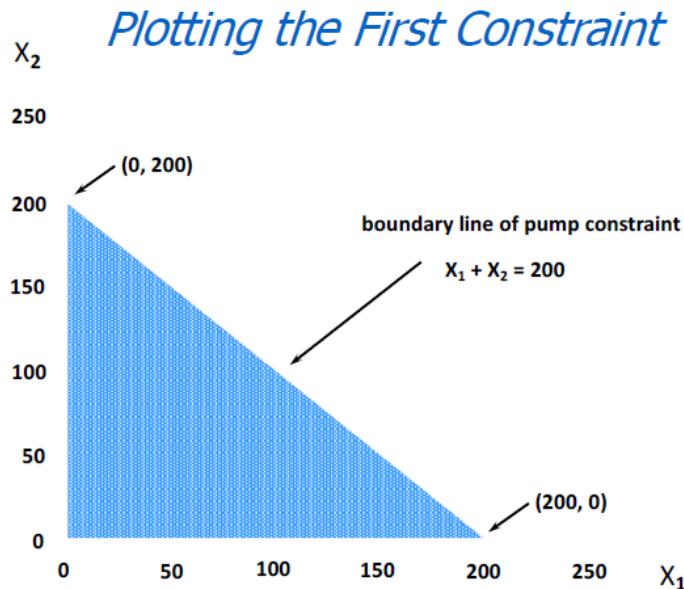


Figure 3.3: Graphical Linear Solution

Tracciate quindi la retta corrispondente al primo vincolo ($X_1 + X_2 = 200$) e consideriamo i punti ammissibili che sono tutti i punti che soddisfano il vincolo, ovvero tutti i punti con $X_1 + X_2 \leq 200$ che sono tutti quelli evidenziati in azzurro.

Plotto il secondo vincolo allo stesso modo del primo:

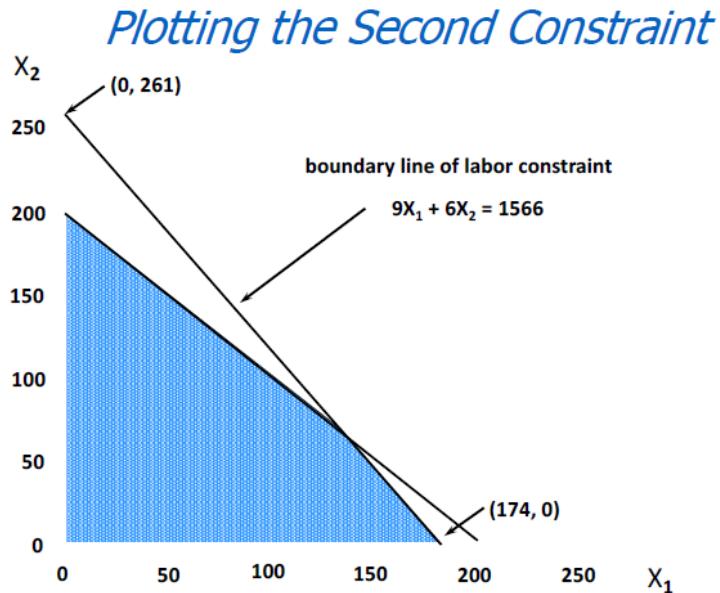


Figure 3.4: Second Constraint

Riducendo ulteriormente la mia regione ammissibile: i punti in blu sono i punti che soddisfano sia il primo che il secondo vincolo. Aggiungo quindi il terzo vincolo:

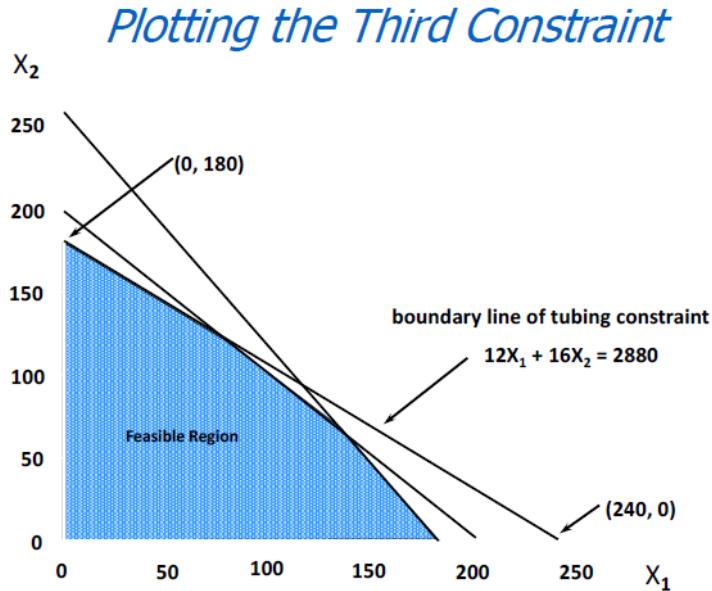


Figure 3.5: Third Constraint

e tengo solo i punti che soddisfano il vincolo che sono quelli al di sotto della retta e la mia regione ammissibile è stata ulteriormente limitata. Otterremo quindi la seguente regione ammissibile:

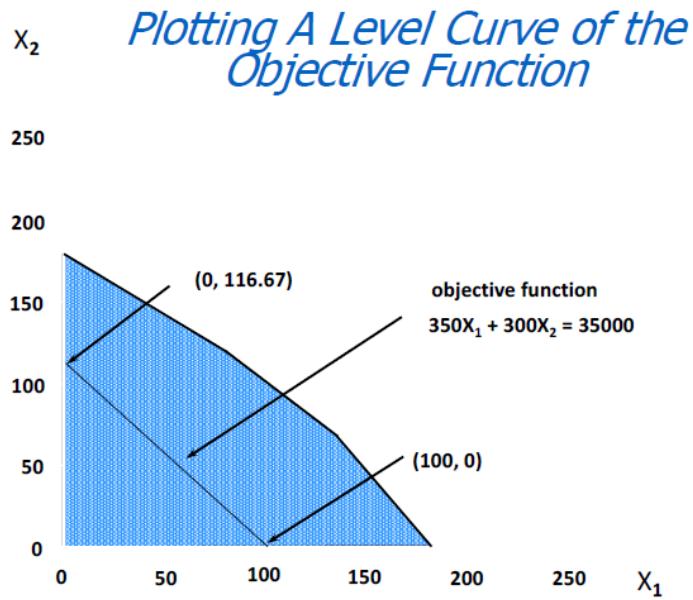


Figure 3.6: Final Feasible Region

Che i punti siano sopra o sotto i vincoli dipende anche dal *segno dei coefficienti*, per cui si ha da stare attenti quando si utilizza una soluzione grafica a prendere il semipiano corretto: quando tutti i coefficienti sono *positivi* il vincolo è di *minore/uguale* e la parte da prendere è quella sottostante i vincoli; ma se i coefficienti fossero *negativi* ci potrebbero essere dei casi in cui la parte da scegliere è quella sopra.

Dopodiché abbiamo la funzione obiettivo che non conosciamo ma conosciamo la sua pendenza, il suo **coefficiente angolare**. Come? Diamo un valore alla funzione obiettivo, disegniamo la retta corrispondente e abbiamo quindi l'inclinazione della retta. Questa retta può diminuire (in questo caso vale 35000): se spostiamo la retta in basso, il valore diminuirà e viceversa. Se vogliamo quindi massimizzare la nostra funzione obiettivo dobbiamo spostare questa retta che rappresenta la funzione obiettivo in alto, come si può vedere:

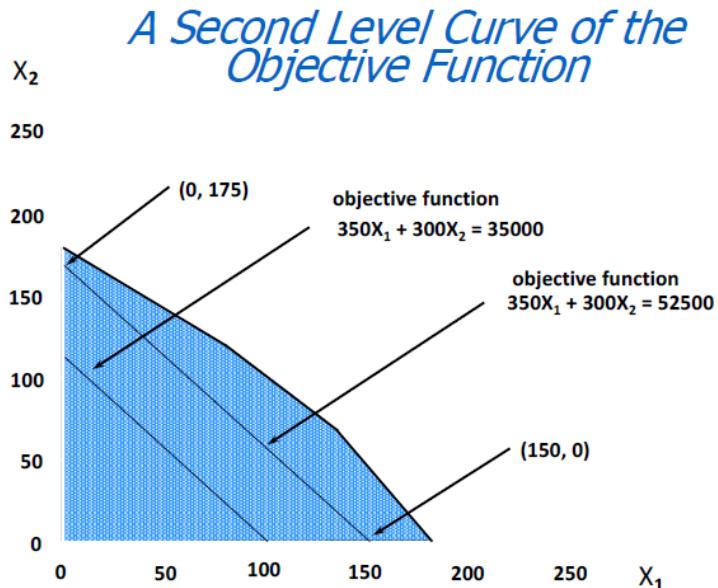


Figure 3.7: Move the objective function

Il nostro obiettivo è quello di spostare la retta il più in alto possibile senza uscire dalla regione ammissibile; voglio trovare quel punto a cui corrisponde quella retta di funzione obiettivo il più alto possibile. Continuo a shiftarla fino a raggiungere il punto della regione ammissibile che ha una più alta funzione obiettivo, ovvero:

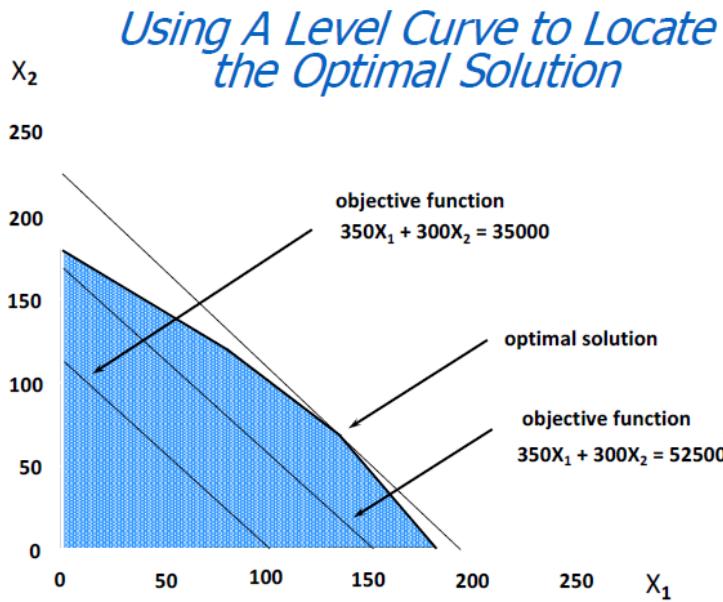


Figure 3.8: Final Solution

Vedo che la soluzione ottimale si trova all'incrocio tra due vincoli che erano l'incrocio tra il vincolo $9X_1 + 6X_2 = 1566$ e $X_1 + X_2 = 200$. Risolvendo un sistema di equazioni in due variabili e trovo il valore del **punto di ottimo**.

Quindi:

La soluzione ottima si verifica nel punto in cui il vincolo sulle *pompe* e il vincolo sulle *ore di lavoro* si intersecano. Abbiamo quindi due equazioni:

$$\begin{aligned} X_1 + X_2 &= 200 \\ 9X_1 + 6X_2 &= 1566 \end{aligned} \tag{3.6}$$

risolviamo le due equazioni e troviamo che la **soluzione ottima** sarà:

$$\begin{aligned} X_1 &= 122 \\ X_2 &= 78 \end{aligned} \quad (3.7)$$

che presenta un **profitto totale** pari a:

$$\text{Total profit} = 350 \times 122 + 300 \times 78 = 66,100 \quad (3.8)$$

profitto più alto di quanto avevamo ottenuto precedentemente.

Quindi vediamo che la soluzione ottimale non era quella di produrre più X_1 possibili solo perché avevano un profitto più elevato. Ma c'è un modo migliore per riutilizzare le risorse e in questo caso tutte le ore di lavoro e le pompe sono utilizzate, per cui le risorse che rimangono inutilizzate sono minori. La soluzione che massimizza il numero di prodotti con maggior profitto non è sempre la soluzione ottimale.

Possiamo vedere diversi punti cosiddetti *Corner Points* che ci dicono quali sarebbero gli altri possibili risultati se producessimo solo di X_1 o solo di X_2 :

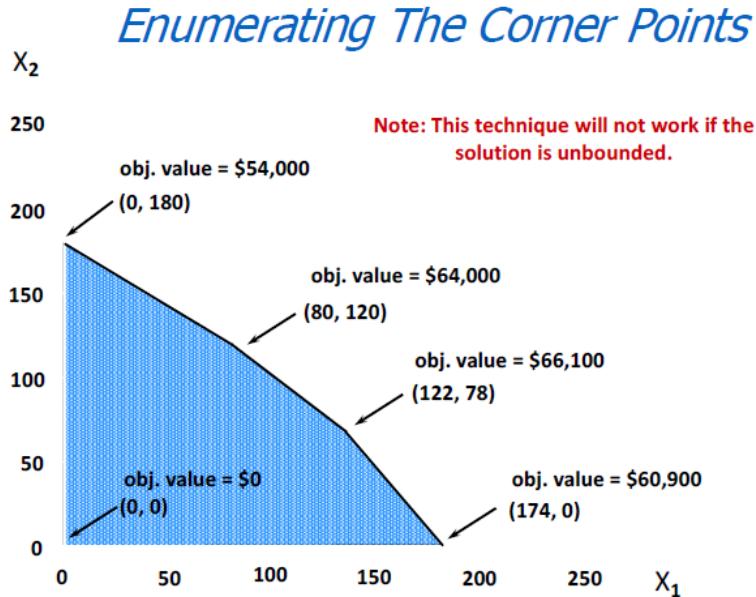


Figure 3.9: Corner Points

rilevando che la nostra soluzione ottima si trova in un vertice che fornisce il profitto più alto possibile. Ovviamente, se la regione ammissibile fosse illimitata potrei far crescere la mia funzione obiettivo a piacere senza trovare però una soluzione. In assenza di vincoli potrei produrre infinite vasche idromassaggio, perché non avrei vincoli in termini di risorse. Questo è un problema, generalmente, di modellazione perché nella realtà sarà difficile trovare un caso in cui possa raggiungere un valore infinito.

A conclusione, per la soluzione grafica del problema si dovrà:

1. Disegnare le rette corrispondenti a ciascun vincolo;
2. Identificare la regione ammissibile;
3. Determinare il punto di ottimo:
 - (a) Disegnando le curve di livello della funzione obiettivo;
 - (b) Numerare i *punti estremi*, ovvero un **vertice**, quindi numerare i vertici e scegliere quello maggiore;

Chapter 4

QUARTA LEZIONE

Problema dell'ottimizzazione: cosa significa ottimizzare in senso matematico? Quando si ottimizza qualcosa stai "creando il meglio". Ma il "meglio" può variare. L'ottimizzazione matematica è quella branca della matematica che ci aiuta a prendere delle decisioni, sia massimizzando che minimizzando degli obiettivi. Il nostro focus di questa prima parte di corso è la modellazione e valutazione delle soluzioni piuttosto che nel metodo risolutivo. Con l'ottimizzazione matematica si trasforma il problema in termini matematici e si sfrutta l'algoritmo per trovare il minimo/massimo. La cosa più importante qual è? Il saper formulare bene il problema o saperlo risolvere bene? Sicuramente devo aver prima formulato meglio il problema.

La parte più difficile è ovviamente la modellazione: bisogna essere a contatto con gli esperti di dominio (medico che valida il modello, nell'ambito logistico chi valuta il modello). Necessaria interazione con gli esperti di dominio altrimenti si rischia di ottimizzare un modello che non serve.

La programmazione matematica è uno strumento che ci aiuta a modellizzare, risolvere dei problemi di ottimizzazione e in generale la programmazione matematica può tenere conto di risorse limitate e può essere **vincolata**, ponendo dei limiti alle nostre decisioni entro il quale deve essere trovata una soluzione ottima.

Limitare lo spazio decisionale, può semplificare o complicare il problema? Dal punto di vista applicativo, inserire dei vincoli limita le possibili soluzioni, ma aumenta la richiesta computazionale. **L'ottimizzazione matematica sono tecniche che cercano l'ottimo di una funzione obiettivo considerando anche dei vincoli.** (Programmazione: programmare delle attività, delle scelte, delle decisioni, non nel senso informatico del termine.)

Quali sono le caratteristiche principali di un problema di ottimizzazione?

- Variabili decisionali; (decisioni da prendere)
- Obiettivo da minimizzare/massimizzare tramite una funzione obiettivo di queste variabili; (decisioni che devo prendere massimizzando/minimizzando questa funzione)
- Soddisfando i vincoli (constraints) che queste variabili devono soddisfare.

La funzione obiettivo rappresenta l'obiettivo del problema che può essere sia massimizzata/minimizzata; le variabili sono sconosciute di cui non sappiamo il valore e vogliamo trovarlo. Bisogna distinguere le **variabili dai parametri** (mentre nel ML le variabili diventano i nostri parametri: problema di linguaggio, devo trovare i valori dei parametri che diventano le nostre variabili). Nel contesto che vedremo noi della programmazione, le variabili sono le leve sulle quali agire per ottimizzare la funzione obiettivo, quindi sono sconosciute e devo determinare il valore ottimo, mentre i parametri sono i dati del modello.

E poi abbiamo una serie di vincoli che limita il valore di queste variabili in un certo spazio. Alla fine il problema di ottimizzazione è trovare il valore delle variabili soddisfando i vincoli.

La forma generale di un problema di ottimizzazione: ho delle variabili di decisioni. Se tutte le funzioni in un problema lineare sono lineari, allora si parla di **Linear programming problem**.

$$\left\{ \begin{array}{l} \text{MAX (OR MIN)} : f_0(X_1, X_2, \dots, X_n) \\ \text{Subject to : } f_1(X_1, X_2, \dots, X_n) \leq b_1 \\ \vdots \\ f_k(X_1, X_2, \dots, X_n) \geq b_k \\ \vdots \\ f_m(X_1, X_2, \dots, X_n) = b_m \end{array} \right. \quad (4.1)$$

ATTENZIONE: se tutte le funzioni in un problema di ottimizzazione sono lineari, il problema è un problema di Programmazione lineare (LP).

Inizieremo a vedere la **programmazione lineare**: noi abbiamo che le funzioni obiettivo e dei vincoli sono tutte funzioni lineari.

$$\left\{ \begin{array}{l} \text{MAX (OR MIN)} : c_1X_1, c_2X_2, \dots, c_nX_n \\ \text{Subject to : } a_{11}X_1, a_{12}X_2, \dots, a_{1n}X_n \leq b_1 \\ \vdots \\ a_{k1}X_1, a_{k2}X_2, \dots, a_{kn}X_n \geq b_k \\ \vdots \\ a_{m1}X_1, a_{m2}X_2, \dots, a_{mn}X_n = b_m \end{array} \right. \quad (4.2)$$

Esempio: LP Problem. Nei problemi di programmazione lineare devono sempre essere \geq , \leq sempre ricordandosi l'uguale! Questo non significa che non possano essere mostrati vincoli di sola minoranza o di sola maggioranza, ma dovremo aggiungere un epsilon aggiuntivo (che sarà da minimizzare o piccolo a piacere). (se impongo un vincolo di uguaglianza impongo che la risorsa sia usata tutta, rendendo probabilmente difficile/impossibile risolvere il problema).

Guardiamo ad un esempio.

Blue Ridge Hot Tubs produces two types of hot tubs: Acqua Spas and Hydro-Luxes.

-	Acqua-Spa	Hydro-Lux
Pumps	1	1
Labor	9 hours	6 hours
Tubing	12 feet	16 feet
Unit Profit	350 dollars	300 dollars

Sono disponibili 200 pompe, 1566 ore di lavoro e 2880 piedi di tubi.

Cinque passi per la formulazione del problema:

1. Capire il problema
2. Identificare le variabili di decisioni (cosa voglio decidere) e come le formalizzo. In questo caso:
 - X_1 = numero di Acqua-Spas da produrre
 - X_2 = numero di Hydro-luxes da produrre
3. Formuliamo la funzione obiettivo in forma lineare, ovvero come combinazione lineare delle variabili decisionali: (la funzione obiettivo è quella di massimizzare il profitto - profitto unitario per la quantità prodotta):

$$\text{MAX : } 350X_1 + 300X_2 \quad (4.3)$$

Dove 350 e 300 sono i profitti.

4. Stabilita la funzione obiettivo bisogna formulare i vincoli, ovvero:

Vincolo sulla disponibilità di **pompe**:

$$1X_1 + 1X_2 \leq 200 \quad (4.4)$$

Vincolo sulla disponibilità di **lavoro**:

$$9X_1 + 6X_2 \leq 1566 \quad (4.5)$$

Vincolo sulla disponibilità di **tubi**:

$$12X_1 + 16X_2 \leq 2880 \quad (4.6)$$

determinando quindi i vincoli che corrispondono alle tre risorse scarse. Perché cos'è che mi limita dal produrre un infinita quantità di queste vasche idromassaggio? Le risorse a disposizione.

5. Identificare i lower bound e gli upper bound eventuali sulle variabili di decisione. Molto spesso il lower bound è di non negatività: la quantità prodotta per ogni vasca idromassaggio dovrà essere maggiore di 0 (non posso produrre meno 4 vasche idromassaggio, la quantità prodotta dovrà essere un numero maggiore/uguale a 0). I vincoli di non negatività (a differenza di ciò che abbiamo detto per cui maggiori sono i vincoli e maggiore la richiesta computazionale) sono vincoli buoni che aiutano dal punto di vista computazione la risoluzione del problema.

$$X_1 \geq 0 \quad X_2 \geq 0 \quad (4.7)$$

Quindi la funzione obiettivo finale del problema sarà:

$$\begin{aligned} & MAX \text{ (OR MIN)} : \quad 350X_1 + 300X_2 \\ & Subject to : 1X_1 + 1X_2 \leq 200 \\ & \quad \quad \quad \vdots \\ & \quad \quad \quad 9X_1 + 6X_2 \leq 1566 \\ & \quad \quad \quad \vdots \\ & \quad \quad \quad 12X_1 + 16X_2 \leq 2880 \\ & \quad \quad \quad X_1 \geq 0 \\ & \quad \quad \quad X_2 \geq 0 \end{aligned} \quad (4.8)$$

Guardando al problema sembrerebbe sensato produrre più X1 possibili avendo un valore del profitto maggiore. Ma quanti sono più X1 possibili? Supponendo che X2 sia uguale a 0, potremmo produrre 200 X1. Nel secondo vincolo si ha che di X1 se ne possono produrre solamente 174 se X2 fosse uguale a 0. Nel terzo vincolo si arriva a 240. Quindi, se X2=0, il valore massimo di X1 è 174 e il profitto totale è dato da:

$$350 * 174 + 300 * 0 = 60900 \quad (4.9)$$

Questa soluzione è ammissibile? (*feasible*, perché soddisfa tutti e tre i vincoli) Certamente! Ma siamo sicuri sia la soluzione ottimale? Rimangono delle risorse inutilizzate! Esiste una combinazione che utilizza meglio le risorse che ho a disposizione?

Ottengo una soluzione **ammissibile** che soddisfa tutti i vincoli del problema, quindi è implementabile. Quindi le risorse che ho sono sufficienti per essere implementate. Ma riusciamo ad ottenere una soluzione migliore? I vincoli del problema di programmazione lineare, definiscono la sua **regione ammissibile**.

In generale:

$$\begin{aligned}
 & \text{MAX (OR MIN)} : c_1X_1, c_2X_2, \dots, c_nX_n \\
 & \text{Subject to : } a_{11}X_1, a_{12}X_2, \dots, a_{1n}X_n \leq b_1 \\
 & \quad \vdots \\
 & \quad a_{k1}X_1, a_{k2}X_2, \dots, a_{kn}X_n \geq b_k \\
 & \quad \vdots \\
 & \quad a_{m1}X_1, a_{m2}X_2, \dots, a_{mn}X_n = b_m
 \end{aligned} \tag{4.10}$$

4.1 Risolvere un problema di programmazione lineare: un approccio grafico

- I vincoli di un problema di programmazione lineare definiscono la sua **regione ammissibile** e il punto della regione ammissibile;
- Il punto della regione ammissibile con la soluzione migliore è la soluzione del problema;
- Per i problemi a due variabili (e solo per i problemi con due variabili) è facile disegnare la regione ammissibile: la soluzione grafica funziona solo in questi casi e normalmente non si usa, avendo molte più variabili-

Quindi, la risoluzione grafica spiega geometricamente cosa succede, perché ci aiuta a comprendere il funzionamento di questi modelli di programmazione lineare e cosa vuol dire trovare il punto di ottimo di questi modelli, non perché si tratta di un metodo largamente utilizzato (perché un problema a due variabili nella realtà sono pochissimi).

Dati i vincoli di non negatività la nostra regione ammissibile è il quadrante con X_1 e X_2 maggiori/uguali a 0 e sarà solo lì che andremo a cercare la nostra soluzione. Dopodiché andremo a tracciare le rette corrispondenti ai vincoli.

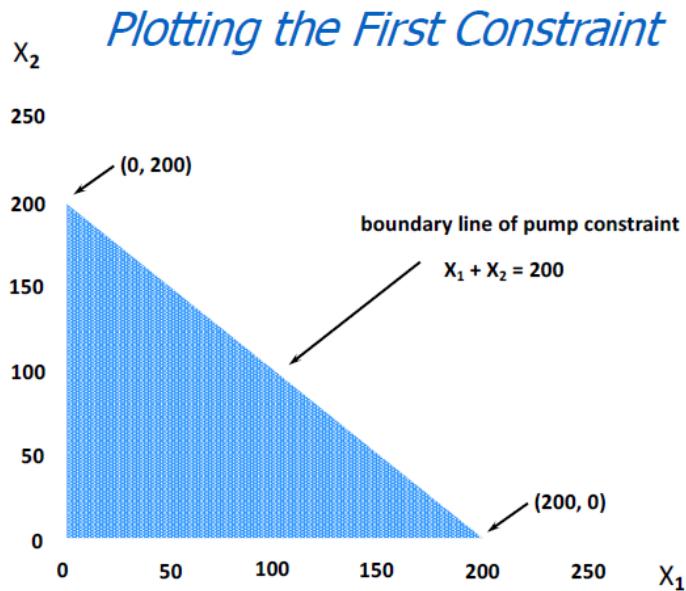


Figure 4.1: Soluzione grafica al problema di PL

Consideriamo i **punti ammissibili**: ovvero tutti i punti che soddisfano il vincolo. Quindi tutti i punti con X_1 e X_2 minore/uguale di 200 che sono tutti quelli evidenziati in azzurro. Dopodiché faccio la stessa cosa con l'altro vincolo:

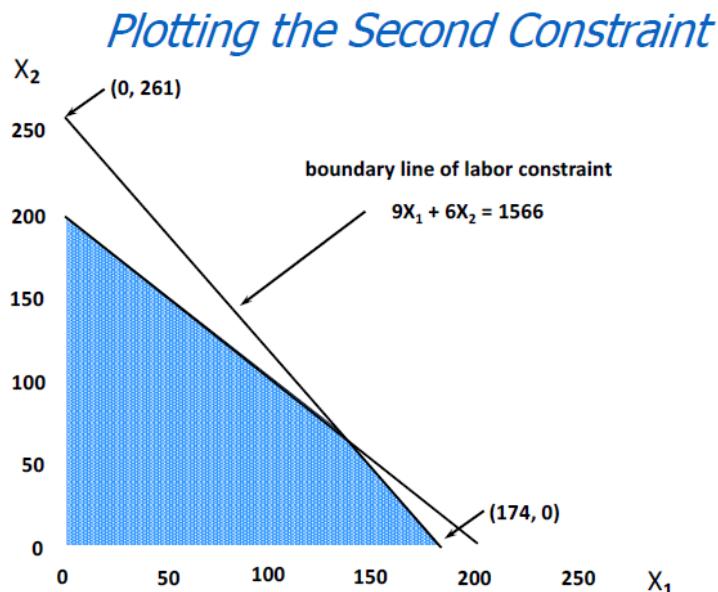


Figure 4.2: Secondo vincolo

riducendo la regione ammissibile. E così anche per il terzo vincolo:

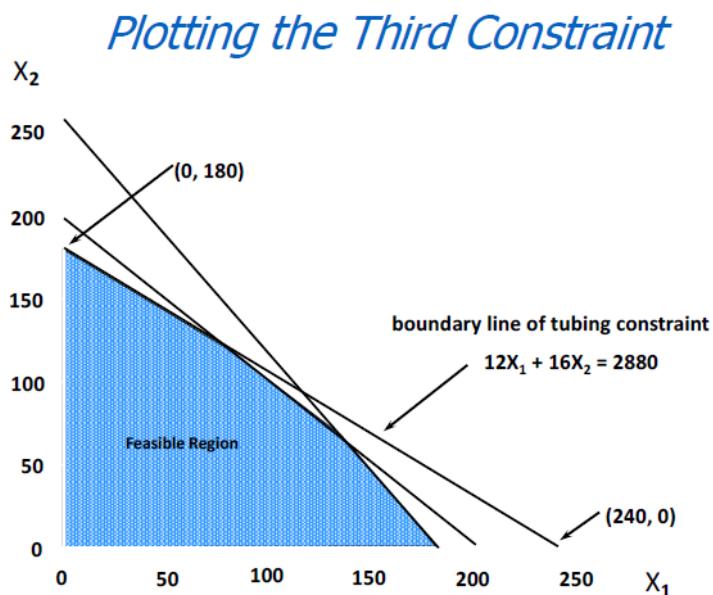


Figure 4.3: Terzo vincolo

Quindi noi non conosciamo la nostra funzione obiettivo ma conosciamo il coefficiente angolare (diamo un valore alla funzione obiettivo e si disegna la retta corrispondente, ottenendone l'inclinazione). L'obiettivo è quello di trovare quel punto a cui corrisponde la retta di funzione obiettivo il più alto possibile, ovvero:

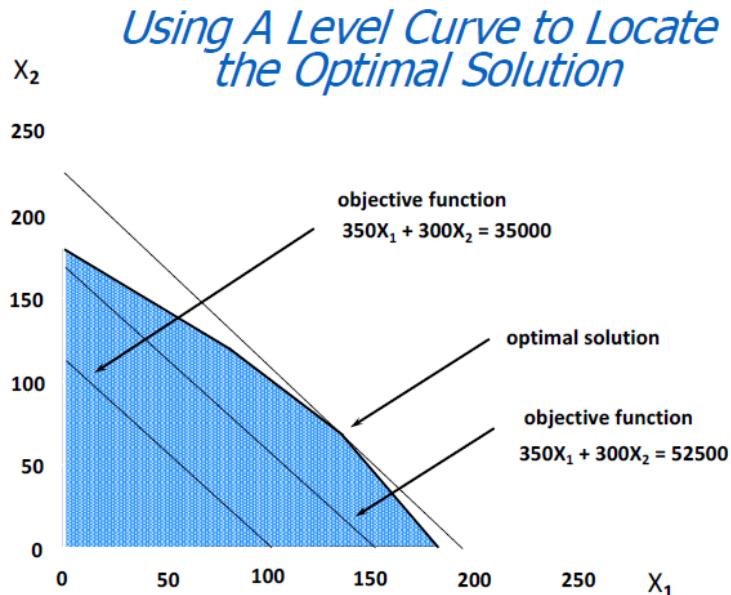


Figure 4.4: Soluzione PL

Come determino il valore del punto ottimo? Il punto si trova all'incrocio tra:

$$9X_1 + 6X_2 = 1566$$

e

$$X_1 + X_2 = 200$$

Si risolve un sistema di due equazioni e si ottiene il punto del valore di ottimo.

La soluzione ottima si verifica quindi nel punto in cui il vincolo sulle pompe e sul lavoro si intersecano:

Calculating the Optimal Solution

- The optimal solution occurs where the “pumps” and “labor” constraints intersect.

- This occurs where:

$$X_1 + X_2 = 200 \quad (1)$$

$$\text{and} \quad 9X_1 + 6X_2 = 1566 \quad (2)$$

- From (1) we have, $X_2 = 200 - X_1$ (3)

- Substituting (3) for X_2 in (2) we have,

$$9X_1 + 6(200 - X_1) = 1566$$

which reduces to $X_1 = 122$

- So the optimal solution is,

$$X_1 = 122, X_2 = 200 - X_1 = 78$$

$$\text{Total Profit} = \$350 * 122 + \$300 * 78 = \$66,100$$

Figure 4.5: Soluzione PL

Profitto più alto che è diverso da quello che avevamo detto inizialmente di produrre quel prodotto con il maggior profitto in assoluto. Si ha un modo migliore di utilizzare le risorse e in questo caso tutte le ore di lavoro e le pompe sono utilizzate.

Questa è la nostra soluzione ottima, si possono vedere anche diversi punti di soluzioni ammissibili ma non ottimali:

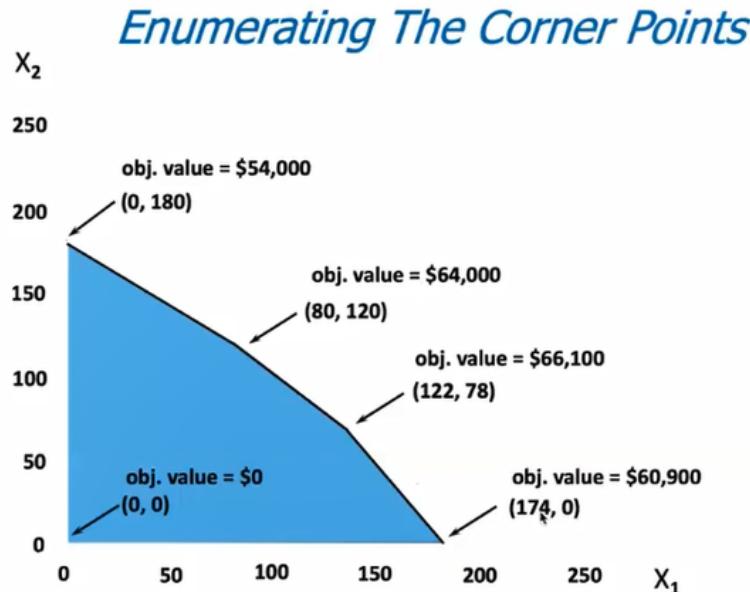


Figure 4.6: Soluzione PL

Questa soluzione funziona solo se abbiamo dei vincoli, altrimenti senza vincoli sulle risorse si ha un problema di modellazione visto che nella realtà non esiste il caso in cui io possa avere le possibilità di costruire infiniti prodotti (la regione ammissibile sarebbe infinita).

Per la risoluzione grafica:

1. Disegnare le rette corrispondenti ad ogni vincolo identificando la regione ammissibile;
2. Identificare la regione ammissibile;
3. Localizzare il punto di ottimo disegnando:
 - le curve di livello della funzione obiettivo;
 - oppure prendere tutti i vertici (la soluzione si trova in almeno un vertice) enumerandoli e scegliere il maggiore.

Chapter 5

QUINTA LEZIONE 17-03-2021

La nostra funzione obiettivo la vediamo in termini economici come la minimizzazione dei costi o massimizzazione del profitto.

Uno degli esercizi dell'esame prevede un testo del problema e noi dobbiamo essere in grado di descrivere il problema mediante un modello matematico.

Guardare esempio di problema di PL in Excel della lezione 4.

La soluzione ottima si può trovare:

- sempre su un vertice
- su due vertici e su tutti i punti fra i due vertici la funzione obiettivo avrà lo stesso valore che è il valore ottimo/massimo che può raggiungere. Questo succede perché cambiando il coefficiente angolare della funzione obiettivo esso diventa uguale al coefficiente angolare del vincolo e vi si sovrappone.

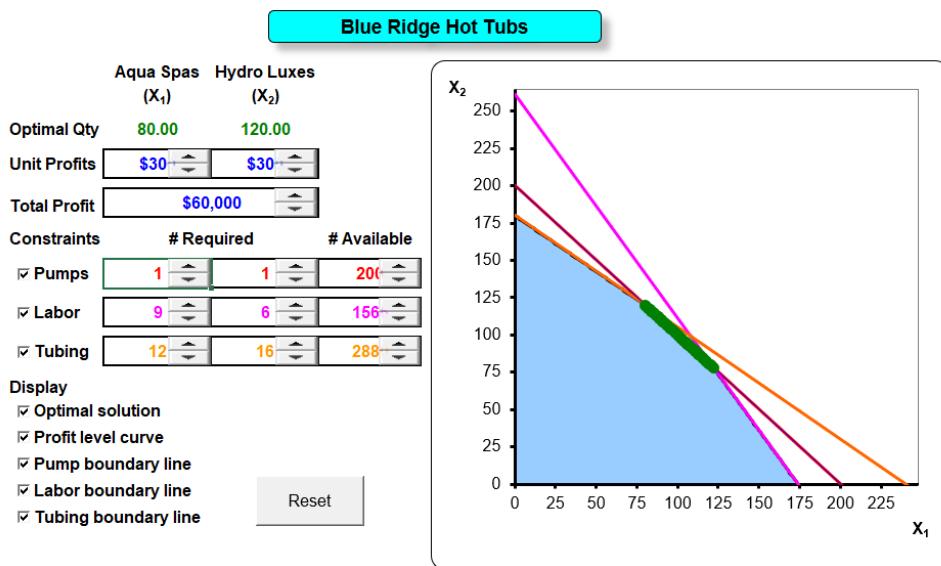


Figure 5.1: Esempio excel

In generale: è meglio avere tanti punti di ottimo oppure uno solo? Se usiamo un modello per fare una scelta ci aspettiamo un'unica risposta, altrimenti con che criterio si sceglie? Seconda cosa: attenzione all'incertezza. Quando si hanno tante soluzioni, è sufficiente un piccolo cambiamento nel coefficiente di una delle variabili per modificare in modo evidente la mia soluzione, oscillando fra una soluzione ottima e l'altra. Se cambio coefficienti della mia funzione obiettivo cambierà la pendenza dei vincoli. Vedremo

meglio questo aspetto quando vedremo l'analisi di sensitività.

Cambiando invece i termini noti, ciò che cambia (aumento le risorse ad esempio) arrivo ad avere risorse in eccesso e diventa ridondante per la soluzione:

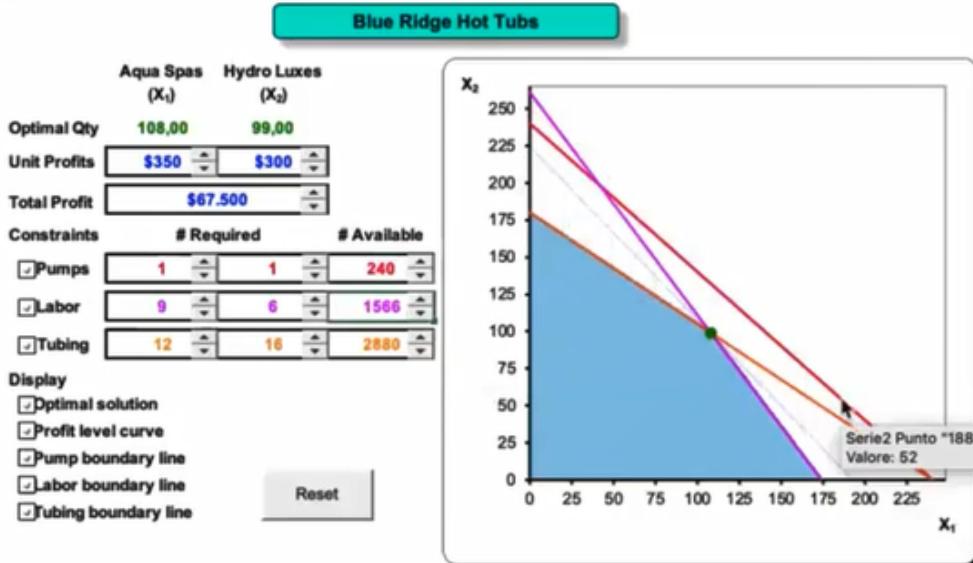


Figure 5.2: Modifica dei termini noti

Quindi, cambiando il termine noto dei vincoli cambia la regione ammissibile (si rilassa o si contrae: se il vincolo diventa più difficile da soddisfare si contrae; si allarga se il vincolo diventa più facile da soddisfare).

Altro aspetto da tenere in considerazione è la possibile presenza di una **regione illimitata**: se non ho un vincolo che mi limita la regione ammissibile potrei avere una regione ammissibile illimitata come nel seguente caso:

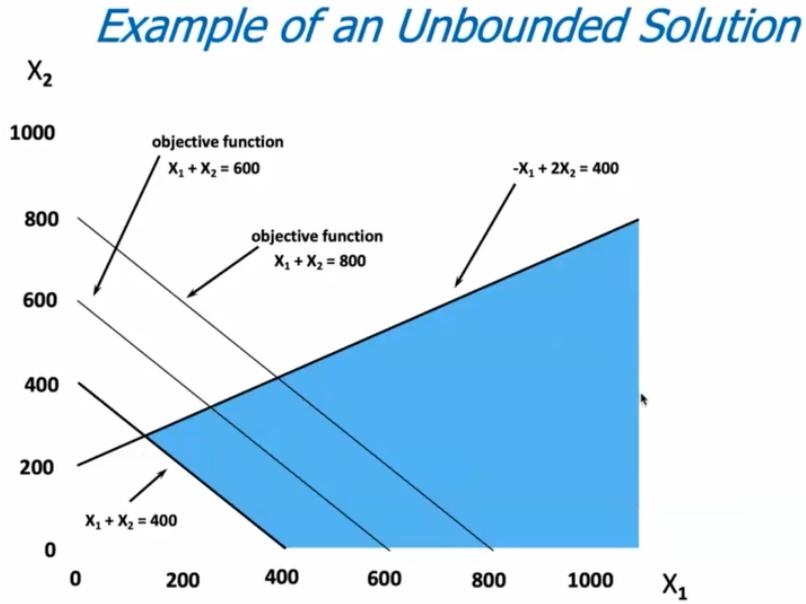


Figure 5.3: Unbounded solution

e la mia funzione obiettivo può crescere all'infinito, non ottenendo una soluzione ottima perché non si ha un vertice che limita la crescita della funzione obiettivo. Quando accade si ha un problema di modellazione o non si è considerato qualche vincolo: è impossibile avere soluzioni illimitate perché non

corrispondono a delle possibili decisioni.

L'altro caso che può accadere è di avere una **unfeasible solution** (problema di non ammissibilità della soluzione): si hanno dei vincoli che non hanno soluzioni in comune e quindi non ci sono soluzioni che soddisfino contemporaneamente tutti i vincoli. (le regioni/i semipiani definiti dai vincoli non hanno punti di intersezione e non esistono soluzioni che soddisfano contemporaneamente tutti i vincoli)

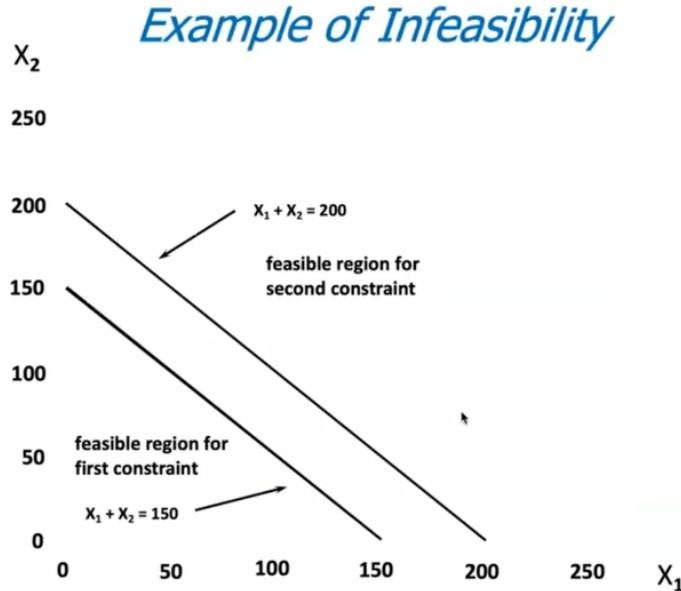


Figure 5.4: Unfeasible solution

Quali sono le **assunzioni fondamentali** per un problema di programmazione lineare sono:

- Proporzionalità: il contributo alla funzione obiettivo da ciascuna variabile decisionale è proporzionale al valore della variabile decisionale. Esempio: il contributo al profitto dal costruire 4 Acqua-Spas (4×350 dollari) è 4 volte il contributo che viene fornito al profitto dal costruire 1 Acqua-Spa (350 dollari);
- Additività: il contributo al valore della funzione obiettivo per ciascuna variabile di decisione è indipendente dai valori delle variabili di decisione. Esempio: non importa il valore di X_2 , la creazione di X_1 contribuirà sempre con 350 dollari alla funzione obiettivo.

Che derivano dalla linealità della funzione obiettivo e dalle funzioni dei vincoli.

Allo stesso modo, poiché ogni vincolo è lineare, valgono le seguenti implicazioni:

- Proporzionalità: il contributo di ciascuna variabile a sinistra del segno di uguaglianza/disuguaglianza di ciascun vincolo è proporzionale al valore della variabile. Esempio: ci vogliono 3 volte tanto di ore di lavoro ($9 \times 3 = 27$ ore) per creare 3 Acqua-Spas come ci si impiega per crearne 1 ($9 \times 1 = 9$ ore);
- Non ci sono economie di scala; (non si possono rappresentare problemi con economie di scala tramite modelli lineari);
- Additività: il contributo di ogni variabile decisionale al lato sinistro di un vincolo è indipendente dai valori di ogni altra variabile decisionale. Esempio: non importa il valore di X_1 (numero di Acqua-Spas prodotte), la produzione di X_2 di Hydro-Luxes sarà pari a:
 - $X_2 = \text{pompe};$
 - $6X_2 = \text{ore di lavoro};$

- 16X2 = piedi di tubi;

Esistono altre assunzioni dietro i problemi di programmazione lineare che sono:

- Assunzione di divisibilità: ogni variabile di decisione può assumere valori frazionari (nei problemi di PL possiamo avere valori frazionari, non c'è garanzia di interezza);
- Assunzione di certezza: quando risolvo un problema di programmazione lineare assumo che i dati siano certi, ovvero non abbiamo variabili aleatorie nel modello (si tratta di un problema di ottimizzazione in condizione di incertezza);

5.0.1 Sensitivity Analysis

Quando risolviamo un problema di PL si assume che i dati siano certi. In realtà, si è visto che alcune incertezze possono esistere, riguardo ai prezzi, riguardo alle risorse etc.

Sarà quindi necessario andare a vedere cosa succede quando si cambia un valore. Ovvero si deve applicare una **analisi di sensitività**: mi permette di sapere cosa succede al problema quando cambio un valore dei miei parametri senza dover risolvere nuovamente il problema. Mi permette di rispondere a certe domande per capire cosa succede se cambio i parametri del problema senza necessariamente dover risolvere con il nuovo valore il problema.

L'analisi di sensitività non ci dice solamente che cosa succede se cambio un valore ma anche che cosa devo fare per migliorare le performance del mio modello.

Da guardare i problemi caricati online, ovvero: (problemi standard della ricerca operativa)

- LP Model Example: A production problem;
- LP Model Example: An Investment problem;
- LP Model Example: A Transportation problem;
- LP Model Example: A Blending Problem;
- Homework Model Formulation 1
- Homework Model Formulation 2

ATTENZIONE: la formulazione del problema è una parte fondamentale e complicata, si acquisisce con l'esercizio e con il tempo. La risoluzione viene svolta dal software, la parte per noi necessaria da sapere è l'interpretazione e modellazione.

Chapter 6

SESTA LEZIONE - Practical Session 1

Partiremo dall'esercizio due sui modelli lineari (Modelli di Linear Programming): nel primo problema dovremo modellare il problema, mentre nel secondo dovremo trovare l'ottimo di un problema di programmazione lineare usando il metodo grafico che è uno degli esercizi che spesso viene dato all'esame. (fare riferimento a *Practical Session 1 - EX. 2 Linear Programming in R - An Introduction*).

In questa prima unità di questo corso ci occuperemo dei modelli lineari dove tutte le funzioni coinvolte (le $f_i(x)$ e le $h_i(x)$) sono lineari e le variabili sono tutte continue e positive, ovvero sono definite in $\mathbb{R}^n \geq 0$.

Per esprimere il modello si può scrivere come segue:

$$\begin{aligned}
 & \text{minimize} && c_1x_1 + c_2x_2 + \cdots + c_nx_n \\
 & \text{Subject to :} && a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\
 & && a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\
 & && \vdots \\
 & && a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \\
 & && x_1, x_2, \dots, x_n \geq 0
 \end{aligned} \tag{6.1}$$

dove con n si intende l' n -esima variabile decisionale. Tutto ciò che è presente a sinistra del simbolo di minore/uguale viene chiamato il **left-hand side**. Alla destra del simbolo di minore/uguale si ha il **right-hand side** (elemento a destra del vincolo) che è il valore che limita le nostre variabili. (da 1 ad m dove con m si intendono m-vincoli). Infine, abbiamo che tutte le variabili decisionale sono tutte positive o nulle. Questo ci permette di scrivere il nostro modello matematico sotto forma matriciale come:

$$\begin{aligned}
 & \min && c^T x \\
 & \text{Subject to :} && Ax \leq b \\
 & && x \geq 0
 \end{aligned} \tag{6.2}$$

La regione ammissibile per un problema lineare quindi definito da vincoli lineari viene chiamata **politopo**: è un termine che descrive un oggetto geometrico che ha delle facce piatte delimitate da segmenti che è una generalizzazione multidimensionale del concetto di poliedro tridimensionale (un poliedro in tre dimensioni è facile da immaginare: un cubo, un dado, un dado con più facce come il dodecaedro). Il politopo ha le stesse caratteristiche ma in uno spazio multidimensionale. Perché è importante saperlo? Perché vorrà dire che lo spazio delle soluzioni ammissibili è delimitato da degli iperpiani e vedremo che questo implicherà che la soluzione ottima, se esiste, si trova su uno di questi iperpiani.

Vediamo ora il problema della linearità dal punto di vista di chi deve modellare un certo problema. Un problema di PL può essere modellato come tale se sono verificate 4 condizioni:

- Condizione di proporzionalità:** ogni variabile ha un effetto sulla funzione obiettivo e sui vincoli proporzionale al suo valore (ovvero, moltiplicato per una costante);

2. **Condizione di indipendenza e addività:** l'effetto di due variabili sulla funzione obiettivo e qualsiasi vincolo è dipendente dal valore della singola variabile, cioè dalle due variabili prese singolarmente ed è un effetto additivo, ovvero se una aumenta di un certo valore, l'impatto sulla funzione obiettivo è pari al delta di quanto è aumentato/diminuito;
3. **Condizione di divisibilità:** ogni variabile decisionale è infinitamente divisibile (stiamo trattando problemi che cadono nel campo dei reali e non degli interi). Una soluzione per cui il valore di una variabile decisionale è una frazione ha senso per noi;
4. **Condizione di certezza:** quando stiamo modellando un certo problema di PL tutti i coefficienti, sia quelli della funzione obiettivo che quelli della matrice dei vincoli, sono certi, sono dei valori; non subiscono l'effetto della stocasticità/della randomicità. Se l'assunzione viene meno compaiono i problemi di programmazione stocastica che hanno ben altre tecniche per essere risolti.

6.0.1 Finding the optimal solution graphically

Cerchiamo quindi di creare il nostro primo problema di PL CONTINUA tramite il seguente esercizio (uno degli esercizi d'esame è simile a questo). Il problema è un problema di produzione, si vogliono produrre due prodotti e avremo le seguenti descrizioni:

- Each tonne of *Product 1* consumes 30 working hours, and each tonne of *Product 2* consumes 20 working hours. The business has a maximum of 2700 working hours for the period considered; (abbiamo quindi 2700 ore di budget per produrre una certa quantità del prodotto 1 che del prodotto 2);
- As for machine hours, each tonne of *Products 1* and *2* consumes 5 and 10 machine hours, respectively. There are 850 machine hours available;
- Each tonne of *Product 1* yields 20 million dollar of profit, while *Product 2* yields 60 million dollar for each tonne sold;
- For technical reasons, the factory must produce a minimum of 95 tonnes in total between both products;
- We need to know how many tonnes of *Product 1* and *2* must be produced to maximize total profit;

Quali sono secondo noi i passi da seguire per provare ad approntare un modello matematico che rappresenta il nostro problema?

- Cercare di capire qual è l'obiettivo che vogliamo massimizzare/minimizzare;
- Ci dobbiamo chiedere, una volta avuto l'obiettivo in mente, come esprimiamo questo obiettivo in termini di variabili decisionali? Perché in base alle variabili decisionali che scelgo il mio modello si può rappresentare in una maniera o in un'altra. Noi vogliamo (in questo caso specifico), massimizzare il profitto totale, dato dai guadagni meno i costi. In questo caso non abbiamo indicazioni sui costi e possiamo immaginare che l'informazione che ci vengono dati sui guadagni siano solo riferite al profitto. Il profitto dipende da due possibili decisioni: la decisione di quanto produco di prodotto 1 e quanto di prodotto 2. A questo punto, ottenuto i valori, moltiplicherò per 20 milioni per il prodotto 1 e per il prodotto 2. Qui si vede la linearità del problema: se ho prodotto 1 pezzo avrò guadagnato 20 milioni, se produco 2 tonnellate guadagno 40 milioni, lo stesso per il prodotto 2. Inoltre vale l'additività vista prima: se produco una tonnellata di uno e una tonnellata dell'altro quello che guadagno sarà la somma dei due profitti.
- Quindi identificate la variabili decisionali (le nostre leve per massimizzare il profitto) sono in questo caso il numero di tonnellate prodotte di prodotto 1 e il numero di tonnellate prodotte di prodotto 2. Poiché non abbiamo vincoli in termini di interezza delle nostre variabili, potremmo anche produrre delle frazioni di variabili. Questo ha senso per molti prodotti: immaginiamo di produrre due varietà di cemento: si può produrre una tonnellata, mezza tonnellata ecc.

- Si definisce la funzione obiettivo;
- Si definiscono i vincoli;
- Si costruisce il modello;

Nel caso specifico del problema si avrà quindi che:

- DECISION VARIABLE: il problema verrà modellato come un problema di programmazione lineare e le variabili di decisione saranno:
 - x_1 : number of tonnes produced and sold of Product 1;
 - x_2 : number of tonnes produced and sold of Product 2;
- OBJECTIVE FUNCTION: The profit coefficients of these variables are 20 and 60, respectively. Therefore, the objective function is defined multiplying each variable by its corresponding coefficient: (si considerano i valori già in milioni e si scrive solo 20 e 60)

$$f_0(x) = 20x_1 + 60x_2 \quad (6.3)$$

- CONSTRAINTS: i vincoli saranno:

- A constraint making that the total amount of working hours used in *Product 1* and *Product 2*, which equals $(30x_1 + 20x_2)$ is less or equal than 2700 hours. (budget). Vincolo di **budget**: ho una risorsa che sono le ore di lavoro umano che devono essere allocate per la produzione di questi due prodotti;
- A similar constraint making that the total machine hours $(5x_1 + 10x_2)$ are less or equal than 850. (budget). Non si può richiedere più ore macchine di quante siano disponibili: vincolo di **budget** perché ho un budget di risorse disponibili che devo allocare cercando di massimizzare il mio profitto;
- A constraint making that the total units produced and sold $(x_1 + x_2)$ are greater or equal than 95. (bootstrap), definito anche **vincolo di bootstrap**, necessario a far partire la produzione (bootstrap = *far progredire*). Bisogna produrre **almeno** 95 tonnellate;

A questo punto si costruisce il modello come:

$$\begin{aligned} & \text{maximize} \quad 20x_1 + 60x_2 \\ \text{Subject to :} \quad & 30x_1 + 20x_2 \leq 2700 \\ & 5x_1 + 10x_2 \leq 850 \\ & x_1 + x_2 \geq 95 \\ & x_1 + x_2 \geq 0 \end{aligned} \quad (6.4)$$

In forma matriciale lo possiamo esprimere come:

$$\begin{aligned} & \text{maximize} \quad (20 \quad 60) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ \text{Subject to :} \quad & \begin{bmatrix} 30 & 20 \\ 5 & 10 \\ -1 & -1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 2700 \\ 850 \\ -95 \end{pmatrix} \\ & \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{aligned} \quad (6.5)$$

Per poterlo esprimere in forma matriciale quindi con tutti i vincoli di minore/uguale abbiamo dovuto moltiplicare per -1 la parte destra e la parte sinistra della terza disequazione (del terzo constraint).

Essenzialmente avendo due variabili possiamo disegnare questi vincoli su uno spazio cartesiano a due dimensioni. Proviamo a capire che forma ha lo spazio delle soluzioni ammissibili:

```
##Finding the optimal solution graphically
```

In the following figure you can see a 2-dimension representation of the considered linear problem. You can see the feasible region in gray color and the level curve for the objective function in red. Since the higher the red line the better, the maximum can be found graphically and is equal to 4900.

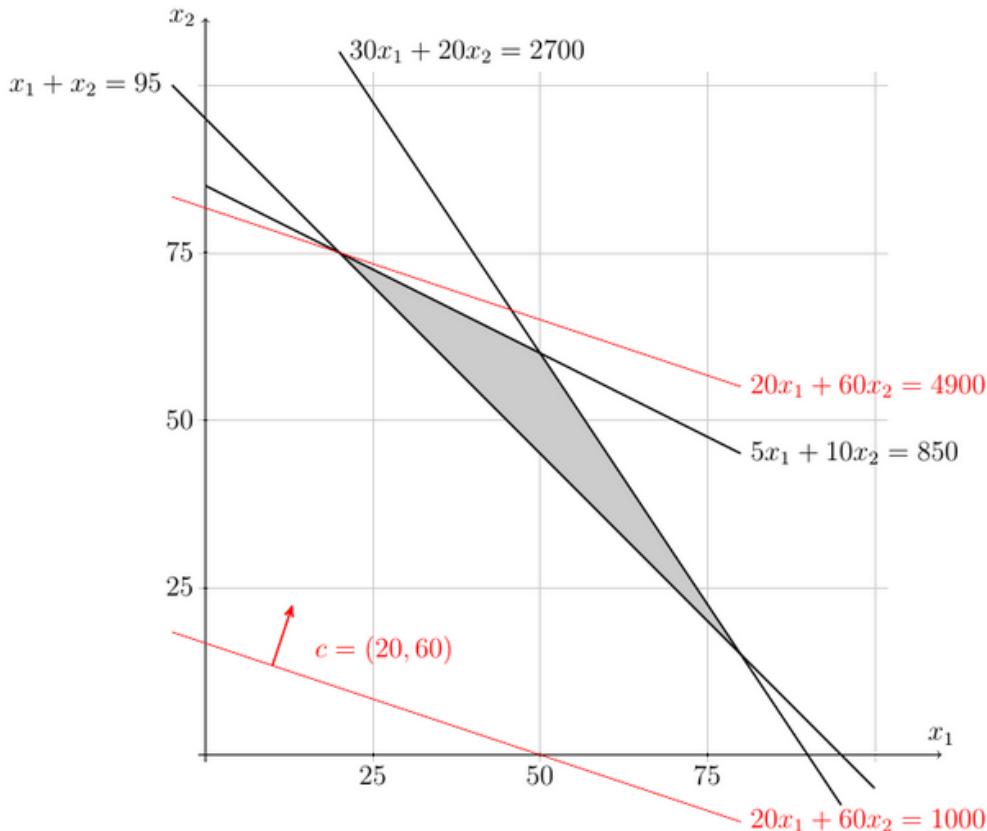


Figure 6.1: Optimal graphical solution

Il luogo per i quali tutti e tre i vincoli sono verificati, ovvero dei punti x_1 e x_2 che verificano tutti e tre i vincoli è costituito dal triangolo rappresentato. Solo le soluzioni appartenenti a questo triangolo sono soluzioni che sono **ammissibili per il problema** e quindi solo una di queste soluzioni che sono infinite è la **soluzione ottima** per il nostro problema.

Imponendo il valore della funzione obiettivo uguale ad una certa costante si ottiene la retta in rosso nella figura. Così facendo avremmo identificato le curve di livello, ovvero l'insieme dei punti dell'iperpiano che definisce la nostra funzione obiettivo che assumono tutti lo stesso valore, in questo caso ad esempio 1000. L'insieme dei punti del dominio che assume quel valore, proiettato sull'asse x_1 e x_2 , rappresenta questa curva di livello che in questo caso è una retta, dato che la nostra funzione obiettivo è lineare e abbiamo un iperpiano e su un iperpiano le curve di livello sono delle rette.

Abbiamo quindi una retta e aumentando/diminuendo il valore della funzione obiettivo, vediamo che la retta man mano che questo valore cresce, si sposta verso l'alto generando una serie di rette tra loro parallele. Questo significa che al crescere della funzione obiettivo questa curva di livello intersecherà il nostro spazio delle soluzioni ammissibili e in particolare lo intersecherà nel vertice a circa 80 di x e 20 di y (il primo punto ammissibile per cui è possibile trovare un valore della funzione obiettivo, che sarà quindi il minimo della nostra funzione obiettivo); e spostandosi verso l'alto si raggiunge il vertice incrociato dall'ultima retta rossa che ha valore pari a 4900. Tale punto è il **punto ottimale** e ci identifica il valore della funzione obiettivo pari a 4900.

Questo procedimento è alla base del cosiddetto del **metodo grafico per la risoluzione dei problemi di PL**.

Da notare che entrambe le soluzioni (sia quelle minima che quella ottima si trovano sui vertici del nostro poligono (in due dimensioni) (poliedro in tre dimensioni) (politopo in n dimensioni)). Questa considerazione non è un caso: la **proprietà fondamentale dei problemi di PL** è che l'ottimo della funzione obiettivo si troverà, sempre se esiste, su uno dei vertici del poliedro che rappresenta lo spazio delle soluzioni ammissibili. Questo non è vero per i problemi più generali di ottimizzazione matematica, come con le funzioni obiettivo quadratiche, dove la soluzione ottima si può trovare anche all'interno della nostra area ammissibile.

6.0.2 Finding the optimal solution analytically

Quindi per trovare una soluzione ottimale ai nostri problemi di PL, sfruttiamo la **proprietà fondamentale dei problemi di PL**:

The optimal solution of an LP (when it exists) can be always found in the extreme points (vertexes) of the polyhedron.

Questo vuol dire che per avere un metodo che ci permetta di risolvere i problemi di ottimizzazione lineare in maniera efficiente dobbiamo avere a disposizione due strumenti:

- Uno strumento che ci permetta di identificare una soluzione iniziale, quindi un vertice del nostro poliedro;
- Una strategia che ci permetta di muoverci da una soluzione iniziale ad un'altra soluzione, cioè da un vertice ad un altro vertice che sia migliore, cioè che abbia un valore migliore di funzione obiettivo.

L'algoritmo che vedremo è chiamato **Algoritmo del Simplex**: identifica le soluzioni all'interno del nostro sistema di equazioni e ci dà una policy che ci permette di passare da un vertice all'altro.

6.0.3 Standard form

Riscriviamo il nostro problema affinché compaia nella cosiddetta **forma standard**:

$$\begin{aligned} & \min c^T x \\ \text{Subject to : } & Ax = b \\ & x \geq 0 \end{aligned} \tag{6.6}$$

I vincoli sono vincoli di **uguaglianza** e non più di minore/uguale. Ciò è possibile farlo se aggiungiamo una serie di variabili che vengono chiamate **variabili di slack** (o variabili di eccesso) che indichiamo con s_i che sono variabili che assumono valori tali da rendere vero il vincolo all'uguaglianza. Esplicitando il nostro modello otterremmo qualcosa di simile:

$$\begin{aligned} & \text{minimize } c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{Subject to : } & a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + s_1 + 0 + \dots 0 = b_1 \\ & a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n + 0 + s_2 + \dots 0 = b_2 \\ & \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n + 0 + 0 + \dots s_m = b_m \\ & x_1, x_2, \dots, x_n, s_1, s_2, \dots, s_m \geq 0 \end{aligned} \tag{6.7}$$

Supponendo quindi che ci siano n variabili e m vincoli (con n possibilmente diverso da m) e abbiamo aggiunto un s_1, s_2, \dots, s_m che devono avere come uguaglianza questi vincoli. Ma i valori che questa variabili assumono, avranno un impatto sulla funzione obiettivo? E' giusto fare questo tipo di trasformazione

o ci cambia il significato del modello? Noi stiamo aggiungendo variabili ai vincoli (le s) e ci chiediamo se queste variabili hanno influenza sulla funzione obiettivo. Poiché non entrano nella funzione obiettivo no, non avranno nessun effetto su di essa. Quindi una trasformazione in forma standard del nostro problema è una trasformazione lecita del nostro problema.

6.0.4 Basic variables

Introduciamo le **variabili di base**: sono una selezione di m variabili che identificano all'interno della matrice dei vincoli precedente una matrice di **rango pieno**: all'inizio avevamo n variabili: x_1, x_2, \dots, x_n , ma avevamo m vincoli con $n > m$. Ora abbiamo aggiunto le variabili di **slack** e la nostra matrice dei vincoli sarà $n + m$. Quello che vogliamo fare è selezionare m di queste variabili (ad esempio: x_1, x_2, s_1) supponendo che m sia uguale a 3) ed estrarre dalla matrice A una matrice di rango pieno, ovvero che il suo determinante è diverso da 0. Queste variabili sono chiamate **variabili di base** (o **basic variables**). A cosa ci serve fare questo? Se impongo che tutte le variabili non di base siano uguale a 0, mi rimane un sistema di equazioni con m variabili ed n vincoli tutti all'uguaglianza e siccome il rango della matrice B è pieno (quindi il determinante è diverso da 0), quel set di equazioni può essere risolto con varie metodologie e identifica un unico punto. Questo vuol dire che facendo in questo modo io ho identificato un unico punto che verifica all'uguaglianza tutti i vincoli che hanno senso. Essenzialmente ciò che sto facendo è identificare l'intersezione tra un certo numero di vincoli. Le variabili di base (o la base delle variabili) è in grado di identificare una soluzione di vertice del mio problema.

Ad esempio, con i dati precedenti:

In our example:

$$\begin{aligned}
 & \text{maximize} && 20x_1 + 60x_2 \\
 & \text{subject to} && \\
 & 30x_1 + 20x_2 + s_1 &=& 2700 \\
 & 5x_1 + 10x_2 + s_2 &=& 850 \\
 & x_1 + x_2 - s_3 &=& 95 \\
 & x_1, x_2, s_1, s_2, s_3 &\geq& 0
 \end{aligned}$$

In matrix form:

$$\begin{aligned}
 & \text{maximize} && (20 \quad 60) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\
 & \text{s.t.} && \begin{bmatrix} 30 & 20 & 1 & 0 & 0 \\ 5 & 10 & 0 & 1 & 0 \\ -1 & -1 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} 2700 \\ 850 \\ -95 \end{pmatrix} \\
 & && x, s \geq 0
 \end{aligned}$$

Extremes points can be found by choosing the set of basic variables:

1. $\{x_1, x_2, s_1\}$

$$\begin{pmatrix} x_1 \\ x_2 \\ s_1 \end{pmatrix} = \begin{bmatrix} 30 & 20 & 1 \\ 5 & 10 & 0 \\ -1 & -1 & 0 \end{bmatrix}^{-1} \begin{pmatrix} 2700 \\ 850 \\ -95 \end{pmatrix} = \begin{pmatrix} 20 \\ 75 \\ 600 \end{pmatrix} \quad s_2 = s_3 = 0$$

2. $\{x_1, x_2, s_2\}$

$$\begin{pmatrix} x_1 \\ x_2 \\ s_2 \end{pmatrix} = \begin{bmatrix} 30 & 20 & 0 \\ 5 & 10 & 1 \\ -1 & -1 & 0 \end{bmatrix}^{-1} \begin{pmatrix} 2700 \\ 850 \\ -95 \end{pmatrix} = \begin{pmatrix} 80 \\ 15 \\ 300 \end{pmatrix} \quad s_1 = s_3 = 0$$

3. $\{x_1, x_2, s_3\}$

$$\begin{pmatrix} x_1 \\ x_2 \\ s_3 \end{pmatrix} = \begin{bmatrix} 30 & 20 & 0 \\ 5 & 10 & 0 \\ -1 & -1 & 1 \end{bmatrix}^{-1} \begin{pmatrix} 2700 \\ 850 \\ -95 \end{pmatrix} = \begin{pmatrix} 50 \\ 60 \\ 15 \end{pmatrix} \quad s_1 = s_2 = 0$$

4. Notice that if we chose $\{x_1, s_2, s_3\}$

$$\begin{pmatrix} x_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{bmatrix} 30 & 0 & 0 \\ 5 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}^{-1} \begin{pmatrix} 2700 \\ 850 \\ -95 \end{pmatrix} = \begin{pmatrix} 90 \\ 400 \\ -5 \end{pmatrix}$$

Which is no feasible and the not an extreme point.

It is easy to verify that all other elections bring to unfeasible solutions. Thus, the optimal solution must be one of the first three points. By computing the value of the objective function we conclude that the solution is the point $x_1 = 20, x_2 = 75, s_1 = s_2 = s_3 = 0$ when $20x_1 + 60x_2 = 4900$ as shown in the figure.

Figure 6.2: Optimal graphical solution

NOTE:

- Siamo andati a sottrarre s_3 perché nel nostro problema originale tale vincolo era di maggiore/uguale;
- Abbiamo 5 variabili ma tre vincoli. Potremo selezionare solo tre variabili per la definizione di una base (avremo diverse scelte come mostrato, supponendo di poter invertire tale matrice). Ottendo poi un vertice con i valori mostrati a destra dell'uguale;

- Se scegliessimo invece s_2, s_3 troveremmo una base non ammissibili (si vede dall'ultimo valore di slack);
- Sono state mostrate solo quelle soluzioni feasible, ovvero con rango pieno.

L'**algoritmo del simplex** si basa quindi sui seguenti passaggi:

- Finding an initial basic solution;
- Explore the closest basic solutions moving in the direction of maximum local increase (if we are dealing with MAXIMIZATION problem) or decrease (in case of a MINIMIZATION problem) of the objective function;
- Stop when an optimal solution is found.

Se l'esempio visto era semplice dal punto di vista del numero delle variabili utilizzate, nella realtà i problemi sono molto più complessi e possono avere decine se non centinaia/migliaia di variabili decisionali. Questi problemi non si risolvono a mano, ma con dei **Solver** che sono dei software che implementano diverse strategie (i solver lineari implementano generalmente l'algoritmo del simplex, particolarmente efficiente per i problemi di PL).

In questa lezione abbiamo anche visto l'"Ex. 4 Linear Programming in R - Graphic method" che è stato dato al primo appello dello scorso anno come esercizio.

ATTENZIONE: in un esercizio simile calcolare tutti i punti viene contato come errore e vengono tolti punti all'esame. Il metodo grafico già ci dà il punto ottimale!

Chapter 7

SETTIMA LEZIONE - Practical Session 2

Guardiamo al problema del **mais**. Supponiamo di essere un agricoltore e di avere 100 acri di terreno alla coltura del mais a disposizione. Abbiamo delle decisioni da prendere e possiamo produrre tre tipi di mais:

- Selling for direct consumption (Corn for food);
- Selling corn to transformation factories for processing into other products (Corn for processing);
- Directly processing into ethanol for fuel and then selling the ethanol (Corn for ethanol);

In base al tipo di mais prodotto ci sarà un costo e un ricavo diversi. Focalizziamoci sull'obiettivo di **massimizzare il profitto** e dobbiamo quindi decidere cosa produrre che dipenderà da quanto costa produrre, infatti:

Costi di produzione per acro:

Corn for Food	125 dollari
Corn for Processing	50 dollari
Corn for Ethanol	150 dollar

Si hanno poi altri dati:

- The farmer only has 10000 dollar available to invest in corn crops (*coltivazione*) (**Vincolo di Budget**);
- Due to the higher standards of marketable fresh corn for consumers, each acre of food-grade corn can only generate 100 bushels (*ceste*) of product per acre, whereas the processing-grade corn can generate 175 bushels (*ceste*) per acre (**Vincolo di Produttività**);
- The fresh corn (Corn for Food and Corn for Processing) must be stored until end of season before it can be dispatched. The storage for fresh corn can only hold 10000 bushels of corn (**Vincolo di Capacità**);
- The generation of ethanol produces 40000 gallons of fuel per acre. The ethanol can be sent out as soon as it is produced so no storage limitations arise (**Vincolo di Produttività**);
- Every acre of land must be allocated to one of the three applications (no unproductive land is permitted);

E i ricavi per unità sono così rappresentati:

Fresh Corn for Food	2.50 dollari/bushel
Corn for Processing	1.50 dollari/bushel
Corn for Ethanol	5.00/1000 gallons

7.0.1 Dissection of the Problem

A questo punto dovremmo avere un'idea, anche vaga, della funzione obiettivo e con questa idea andare a definire:

- Variabili decisionali;
- Con le variabili decisionali, definire completamente la funzione obiettivo;
- Definire i vincoli;
- Se sono presenti, stabilire i vincoli sulle variabili che definiscono gli estremi di validità delle variabili decisionali che dobbiamo considerare;

Quindi il problema è un problema di massimizzazione del profitto il quale viene espresso come i ricavi meno i costi e i ricavi vanno calcolati in base a quanto ogni campo produce. Detto questo, qual è il set di variabili che maggiormente ci sembra adatto per questo tipo di problema? Ovvero, per le variabili, dobbiamo rispondere alla domanda *Quali decisioni devo prendere?*.

Potremmo prendere in considerazione come variabili decisionali quelle relativa alla *quantità di terra allocata al campo per ciascuna produzione*

- x_1 = number of acres used for fresh corn;
- x_2 = number of acres used for processed corn;
- x_3 = number of acres used for ethanol corn;

Volendo massimizzare il guadagno sappiamo esattamente come sarà fatta la nostra funzione obiettivo e se x_1, x_2, x_3 sono gli acri adibiti alla produzione del mais di tipo 1, 2 e 3, quello che dobbiamo fare è moltiplicare la produttività (produciamo 100 bushels per acro e lo vendiamo a 2.50 dollari) e devo sottrarre il costo di produzione. La stessa cosa per il mais di tipo 2 e per quello di tipo 3 (stiamo supponendo che la domanda regga la produzione, ovvero che acquisti interamente il nostro prodotto). Ottenendo quanto segue:

$$P = (2.50 \times 100 - 125) \times x_1 + (1.50 \times 175 - 50) \times x_2 + (5.00 \times 40 - 150) \times x_3 = \\ 125 \times x_1 + 212.5 \times x_2 + 50 \times x_3 \quad (7.1)$$

I vincoli da implementare invece saranno:

Il costo di investimento deve essere inferiore o uguale al vincolo di budget:

$$125 \times x_1 + 50 \times x_2 + 150 \times x_3 \leq 10000 \quad (7.2)$$

Il mais per cibo deve essere immagazzinato e deve rimanere all'interno della capacità di bushels della fattoria:

$$100 \times x_1 + 175 \times x_2 \leq 10000 \quad (7.3)$$

infine, ogni acro di terra deve essere allocato ad almeno una produzione:

$$x_1 + x_2 + x_3 = 100 \quad (7.4)$$

ATTENZIONE: l'identificazione dei vincoli non è sempre immediata e semplice. Per capire meglio come affrontare l'esame porsi la seguente domanda: ho tenuto conto di tutte le affermazioni contenute nel testo del problema? Se sì, si può essere più o meno sicuri di aver creato tutti i vincoli necessari.

Infine, dobbiamo includere il vincolo di **boundaries** che ci dice che:

$$0 \leq x_1, x_2, x_3 \leq 100 \quad (7.5)$$

ovvero, che per x_1, x_2, x_3 sono chiaramente il numero totale di acri che possono essere allocati alle tre variabili. Devono essere maggiori di 0 e minori del numero totale di possibili acri sfruttabili, ovvero 100. Per definire correttamente il modello deve essere messo per forza? In questo caso no, dato che avevamo già definito che x_1, x_2, x_3 dovevano essere uguali a 100 per cui l'unica cosa importante era definire la loro positività.

Quindi, mettendo insieme la funzione obiettivo e i vincoli si ottiene il seguente modello:

$$\begin{aligned}
 & \text{maximize} \quad 125x_1 + 212.5x_2 + 50x_3 \\
 & \text{Subject to :} \quad 125x_1 + 50x_2 + 150x_3 \leq 10000 \\
 & \quad 100x_1 + 175x_2 + 0x_3 \leq 10000 \\
 & \quad x_1 + x_2 + x_3 = 100 \\
 & \quad x_1, x_2, x_3 \geq 0 \\
 & \quad x_1, x_2, x_3 \leq 0
 \end{aligned} \tag{7.6}$$

Proviamo quindi a vedere quali sono i valori effettivi del nostro modello, cioè quanto allochiamo di ogni nostro territorio alla produzione. Utilizzeremo l'API *lpSolveAPI*, come visto nell'ultimo esercizio. (riguardare da *Ex. 3 Linear Programming in R*).

Guardiamo ora all'*Ex. 5 Linear Programming in R* e guardiamo solo alla modellazione del problema.

Abbiamo di fronte un problema di **mixing produttivo** (non è stato mai dato all'esame ma è un tipico problema che compare: uno degli esercizi dell'esame è quasi sempre un problema di modellazione, sia esso continuo o intero o misto)

Si hanno due bibite che vengono create miscelando insieme tre tipi di succhi: pompelmo, pesca e ananas. Per produrre:

- per 10 litri del drink 1, abbiamo bisogno di 1.5 litri di succo di pompelmo, 1 litro di succo di pesca e 0.3 litri di succo di ananas (il resto si immagina sia acqua non contemplata nel modello);
- per 10 litri del drink 2, abbiamo bisogno di 1 litro di succo di pompelmo, 1 litro di succo di pesca, 0.5 litri di succo di ananas.

Anche in questo caso abbiamo un budget disponibile per ognuno dei succhi in produzione:

- Si hanno 2700 litri di succo di pompelmo;
- Si hanno 2100 litri di succo di pesca;
- Si hanno 900 litri di succo d'arancia.

Sappiamo come vengono prodotti i prodotti finali a partire dai semilavorati. Sappiamo la quantità di semilavorati a disposizione e vediamo ora qual è la nostra revenue unitaria per decalitro dei singoli prodotti, supponendo che vengano venduti completamente:

- 13 euro di ricavo per ogni decalitro della bibita 1;
- 10 euro di ricavo per ogni decalitro della bibita 2.

I costi di produzione (sempre per decalitro, quindi ogni 10 litri) sono:

- 5 euro per la bibita 1;
- 3 euro per la bibita 2.

7.0.2 Dissection of the problem

Prima di procedere si devono quindi tenere in considerazione i seguenti elementi:

1. Individuare le variabili decisionali;
2. Definire la funzione obiettivo;
3. Settare i vincoli;
4. Definire le variabili di boundaries;

Quindi:

Le variabili decisionali sono l'ammontare (in decalitri) di ciascun drink da produrre (denotiamo queste variabili come x_1 e x_2): (assumiamo la continuità: possiamo produrre 0.5 decalitri, 1.5 decalitri etc. Non sempre questo è vero! Altrimenti parleremo di modelli di utilizzazione intera)

- x_1 : numero di decalitri del drink uno da produrre;
- x_2 : numero di decalitri del drink due da produrre;

La funzione obiettivo che sarà la differenza tra le entrate e le uscite. In questo caso vogliamo massimizzare i profitti (P) che sarà funzione delle due variabili precedenti, quindi:

$$P = (13 - 5) \times x_1 + (10 - 3) \times x_2 = 8x_1 + 7x_2 \quad (7.7)$$

I vincoli invece saranno pari a:

Nessuna *feasible solution* potrà usare più di 2700 litri di pompelmo, ovvero:

$$1.5x_1 + x_2 \leq 2700 \quad (7.8)$$

Allo stesso modo per il succo di pesca con tetto massimo di 2100:

$$x_1 + x_2 \leq 2100 \quad (7.9)$$

Infine, nessuna *feasible solution* potrà usare più di 900 litri per il succo d'ananas:

$$0.3x_1 + 0.5x_2 \leq 900 \quad (7.10)$$

I *boundaries* terranno conto che x_1 e x_2 dovranno essere maggiori/uguali di 0.
Il modello completo sarà pari a:

$$\begin{aligned} & \text{maximize} && 8x_1 + 7x_2 \\ & \text{Subject to :} && 1.5x_1 + x_2 \leq 2700 \\ & && x_1 + x_2 \leq 2100 \\ & && 0.3x_1 + 0.5x_2 \leq 900 \\ & && x_1, x_2 \geq 0 \end{aligned} \quad (7.11)$$

E si risolve in R oppure con metodo grafico (avendo solo due variabili) come compito a casa.

Chapter 8

LEZIONE 6 - L'algoritmo del Simplex

L'algoritmo del Simplex viene utilizzato per risolvere problemi di programmazione lineare. Abbiamo visto la soluzione grafica che funziona solo con problemi a due variabili, ovvero molto semplici.

Il suo funzionamento:

- Trasforma tutti i vincoli di disegualanza in vincoli di uguaglianza aggiungendo le variabili di **slack**: sono quelle che *assorbono* la differenza tra il termine di sinistra e il termine di destra del vincolo. Se il vincolo è di **minore/uguale** aggiungo una variabile di slack a sinistra e la sommo, in modo che essa assorba la differenza fra il vincolo e il termine noto e quindi trasformi il vincolo di disegualanza in un vincolo di uguaglianza. Analogamente se il vincolo è di **maggiore/uguale** sottraggo una variabile slack che, in questo caso, viene definita di **surplus**. Infatti:

- To use the simplex method, we first convert all inequalities to equalities by adding slack variables to \leq constraints and subtracting slack variables from \geq constraints.

For example: $a_{k1}X_1 + a_{k2}X_2 + \dots + a_{kn}X_n \leq b_k$
converts to: $a_{k1}X_1 + a_{k2}X_2 + \dots + a_{kn}X_n + S_k = b_k$

And: $a_{k1}X_1 + a_{k2}X_2 + \dots + a_{kn}X_n \geq b_k$
converts to: $a_{k1}X_1 + a_{k2}X_2 + \dots + a_{kn}X_n - S_k = b_k$

Note: slack variables S with coefficient -1 are also called Surplus variable

Figure 8.1: Algoritmo del Simplex

Le variabili slack sono importanti perché quando avremo la soluzione esse ci diranno se il vincolo è **attivo**, ovvero se la soluzione ottimale giace su quel vincolo, oppure se il vincolo **non è attivo**, ovvero se non giace su quel vincolo. Questo è fondamentale per fare le analisi di post ottimalità: quando diremo ad esempio che se voglio aumentare il profitto del mio problema, quale risorsa devo mettere a disposizione? Sapendo che la soluzione ottima è vincolata dai vincoli che sono **attivi** andrò ad individuare i vincoli attivi andando ad individuare quei vincoli la cui slack è uguale a 0. Se la soluzione ottima giace su quel vincolo, la variabile di slack sarà ovviamente uguale a 0. Mentre una variabile di slack o di surplus diversa da 0 mi dice che la soluzione non giace su quel vincolo. Quindi se una variabile è uguale a 0 significa che utilizzo tutte le risorse.

Tornando al nostro problema, se ci sono n variabili ed m vincoli in un problema di PL dove $n \geq m$, noi possiamo selezionare m variabili di queste n e risolvere un sistema di equazioni. Come selezioniamo queste m variabili? Selezioniamo queste m variabili e le restanti $n - m$ variabili le poniamo uguali a 0.

$$\begin{aligned}
 & \text{maximize} && 350X_1 + 300X_2 \quad (\text{profit}) \\
 & \text{Subject to :} && 1X_1 + 1X_2 + S_1 = 200 \quad (\text{pumps}) \\
 & && 9X_1 + 6X_2 + S_2 = 1566 \quad (\text{labor}) \\
 & && 12X_1 + 16X_2 + S_3 = 2880 \quad (\text{tubing}) \\
 & && X_1, X_2, S_1, S_2, S_3 \geq 0 \quad (\text{non negativity})
 \end{aligned} \tag{8.1}$$

Quindi se abbiamo tutti i vincoli di disegualanza aggiungiamo una variabile di slack. Ragioniamo sul caso più semplice con il vincolo di minore/uguale. (il vincolo di maggiore/uguale basta moltiplicare per -1). Vediamo cosa succede:

	Basic Variables	Nonbasic Variables	Solution	Objective Value
1	S_1, S_2, S_3	X_1, X_2	$X_1=0, X_2=0, S_1=200, S_2=1566, S_3=2880$	0
2	X_1, S_1, S_3	X_2, S_2	$X_1=174, X_2=0, S_1=26, S_2=0, S_3=792$	60,900
3	X_1, X_2, S_3	S_1, S_2	$X_1=122, X_2=78, S_1=0, S_2=0, S_3=168$	66,100
4	X_1, X_2, S_2	S_1, S_3	$X_1=80, X_2=120, S_1=0, S_2=126, S_3=0$	64,000
5	X_2, S_1, S_2	X_1, S_3	$X_1=0, X_2=180, S_1=20, S_2=486, S_3=0$	54,000
6*	X_1, X_2, S_1	S_2, S_3	$X_1=108, X_2=99, S_1=-7, S_2=0, S_3=0$	67,500
7*	X_1, S_1, S_2	X_2, S_3	$X_1=240, X_2=0, S_1=-40, S_2=-594, S_3=0$	84,000
8*	X_1, S_2, S_3	X_2, S_1	$X_1=200, X_2=0, S_1=0, S_2=-234, S_3=480$	70,000
9*	X_2, S_1, S_3	X_1, S_1	$X_1=0, X_2=200, S_1=0, S_2=366, S_3=-320$	60,000
10*	X_2, S_1, S_3	X_1, S_2	$X_1=0, X_2=261, S_1=-61, S_2=0, S_3=-1296$	78,300

* denotes infeasible solutions

Figure 8.2: Possible Basic Feasible Solutions

Cosa succede? Nel nostro esempio abbiamo 5 variabili ($n = X_1, X_2, S_1, S_2, S_3$). Quindi il numero n di variabili si riferisce al problema aumentato con le variabili di slack. Abbiamo quindi 5 variabili e 3 vincoli. Quindi per individuare i vertici della regione ammissibile possiamo selezionare 3 variabili (che sono uguali al numero di vincoli) e porre le rimanenti variabili uguali a 0.

Di solito partendo da un problema come il nostro si pongono le variabili x_1 e x_2 uguale a 0 e la soluzione sarà $S_1 = 200$, $S_2 = 1566$ e $S_3 = 2880$. E questa equivale ad una soluzione che chiameremo **Soluzione di base**. Quindi tutte le soluzioni che otterremo in questo modo, selezionando 3 variabili e ponendo due variabili uguali a 0 sono dette **soluzioni di base**. Le variabili selezionate sono definite **variabili di base** mentre le variabili che si pongono uguali a 0 sono dette variabili non di base.

La **soluzione ottima** è una **soluzione di base**: quindi significa che per trovare la soluzione ottima devo identificare le **soluzioni di base**. Nella tabella precedente sono state selezionate tutte le soluzioni di base.

Basic Feasible Solutions & Extreme Points

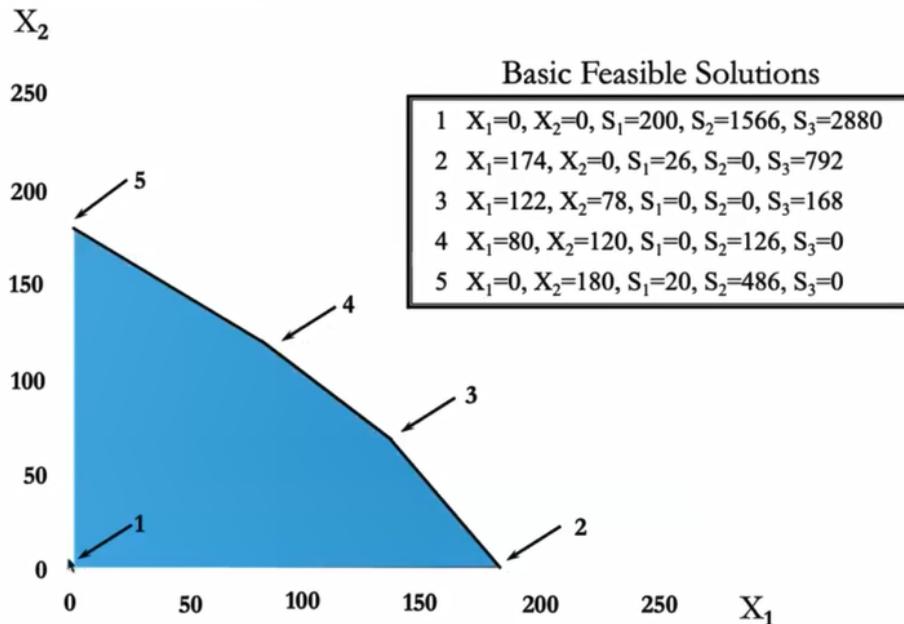


Figure 8.3: Regione ammissibile

Cambiando quindi le variabili di base o non di base ci si sposta sui vertici della regione ammissibile. Cosa si osserva inoltre? Ci sono delle soluzioni che hanno in comune un vertice. Per sapere se due soluzioni sono adiacenti, cioè se hanno un vertice in comune, basta osservare che tutte le **variabili non di base** siano le stesse tranne una: ad esempio nella soluzione 1 si avevano le variabili non di base X_1 e X_2 ; nella soluzione 2 abbiamo che le variabili non di base sono X_2 e S_2 . Quindi sono le stesse tranne una. Nell'algoritmo del simplex questo passaggio si fa andando a sostituire la variabile più conveniente da sostituire, ovvero quella che permette una maggiore crescita della funzione obiettivo. Questo è ciò che fa l'algoritmo del simplex: facendo entrare nelle variabili di base una variabile non di base e facendo uscire una delle variabili di base, cambia il valore della funzione obiettivo e se aumenta si terrà la nuova soluzione.

L'algoritmo del simplex quindi passa da una soluzione di base ad una soluzione non di base mettendo in atto un meccanismo che garantisca la crescita (se sto massimizzando) o la decrescita (se sto minimizzando) della funzione obiettivo.

Risolvendo graficamente il problema abbiamo visto che le soluzioni ottimali del problema stanno sempre su un vertice e nel caso in cui ci siano più soluzioni ottimali significa che la soluzione sta su più vertici perché la funzione obiettivo sarà parallela a qualche vincolo: l'idea dell'algoritmo è quella di individuare questi vertici e di riuscire a passare da un vertice all'altro per cercare la soluzione a questi vertici. L'algoritmo si fermerà quando non ci saranno più direzioni di miglioramento della funzione obiettivo, ovvero quando si sarà arrivati ad una soluzione di base per cui il meccanismo di far entrare una nuova variabile in base e facendone uscire un'altra non produrrà in qualsiasi modo un aumento (se sto massimizzando) o una diminuzione (se sto minimizzando) della funzione obiettivo.

Abbiamo però un problema: le precedenti 5 soluzioni corrispondono alle **soluzioni di base ammissibili** (quindi che giacciono sulla regione ammissibile) del nostro problema. Ma se pongo $S_2 = 0$ e $S_3 = 0$ uguali a 0, ottengo la soluzione $X_1 = 108, X_2 = 99$ e avremo una soluzione che non è sui vertici della regione ammissibile, ma addirittura si troverà al di fuori della regione ammissibile e così anche per la soluzione $7/8/9/10$.

Quindi, ad esempio, $S_2 = 0$ e $S_3 = 0$ significa che il vincolo 2 e il vincolo 3 saranno attivi. La

soluzione 6 ad esempio si troverà nel punto di intersezione tra il vincolo 2 e il vincolo 3 che però sarà fuori dalla regione ammissibile. Come riconosco queste soluzioni per evitare di selezionarle durante le varie iterazioni dell'algoritmo? Almeno una delle variabili diventa **negativa**: se una delle variabili è **negativa**, ponendo $n - m$ variabili uguali a 0 e risolvendo il sistema di equazioni si trova una delle variabili negativa, significa che ci troviamo **fuori dalla regione ammissibile**. E' quindi molto facile individuare una soluzione di base che non sia ammissibile, perché avrà una o più variabili con valore negativo.

Le soluzioni di base ammissibili sono quelle che hanno tutti i valori positivi. Per questo motivo i vincoli di **non negatività** sono **importanti**: quando imposto il problema di programmazione lineare per farlo risolvere all'algoritmo del simplex devo fare in modo che il mio modello soddisfi le condizioni di non negatività. Eventualmente se ho delle variabili che nel problema possono anche assumere valore negativo, posso operare delle trasformazioni in modo tale da avere delle variabili sempre positive. Il problema che arriva come input al risolutore deve soddisfare vincoli di non negatività (ma questo è sempre possibile operando delle trasformazioni).

Quindi l'algoritmo del simplex passa da una variabile di base ammissibile ad un'altra variabile di base ammissibile adiacente. Come fa a passare ad una variabile di base adiacente? Vede quali sono tutte quelle soluzioni che ottengo andando a sostituire una variabile di base con una non di base e facendo uscire una variabile dalla base, ottenendo le possibili soluzioni adiacenti e scelgo quella che mi fa crescere di più la funzione obiettivo (se sto massimizzando) o che me la fa decrescere di più se sto minimizzando.

Quindi il metodo del simplex:

- Individua tutte le soluzioni di base ammissibili che sono sui vertici della regione ammissibile di un problema di PL e si muove nei punti adiacenti andando ad aumentare il valore della funzione obiettivo. Quando raggiunge una iterazione nella quale questa sostituzione non porta più ad un miglioramento significa che ha raggiunto un punto di ottimo;
- Per muoversi da un vertice ad uno adiacente possiamo sostituire una variabile di base con una non di base per creare una nuova soluzione di base ammissibile e mi fermo quando non miglioro più la funzione obiettivo;

8.1 Sensitivity Analysis

Una parte importante della PL è l'**analisi di sensitività**.

Quando risolviamo un problema di PL assumiamo che tutti i coefficienti siano certi: in realtà i coefficienti potrebbero variare per cause esterne oppure vorrei capire cosa succederebbe al mio modello se vario un coefficiente, ovvero se vario la disponibilità di una o più risorse (se aggiungo un tot di una risorsa, come varia la mia funzione obiettivo? Qual è il beneficio che ne ottengo?). Voglio quindi rispondere a domande su cosa accade se cambio il coefficiente del mio problema. Un'importante vantaggio della PL è che non mi richiede di risolvere nuovamente il problema: nella PL utilizzando il metodo del simplex ottengo delle informazioni utili per fare un'analisi approfondita del problema oltre che ottenere una soluzione, potendoci quindi ragionare sopra. L'analisi di sensitività cerca di rispondere alla seguente domanda: quanto è sensibile la soluzione al cambiamento dei coefficienti? (coefficienti della funzione obiettivo e termini noti)

$$\begin{aligned}
 & \text{MAX (or MIN): } c_1X_1 + c_2X_2 + \dots + c_nX_n \\
 & \text{Subject to: } a_{11}X_1 + a_{12}X_2 + \dots + a_{1n}X_n \leq b_1 \\
 & \quad \vdots \\
 & \quad a_{k1}X_1 + a_{k2}X_2 + \dots + a_{kn}X_n \leq b_k \\
 & \quad \vdots \\
 & \quad a_{m1}X_1 + a_{m2}X_2 + \dots + a_{mn}X_n = b_m
 \end{aligned}$$

- How sensitive is a solution to changes in the c_i , a_{ij} , and b_i ?

Figure 8.4: Sensitivity analysis

L'idea alla base della Sensitivity Analysis è quella di dire: di quanto posso cambiare la mia funzione obiettivo mantenendo la soluzione attuale come soluzione ottima? Questo per rispondere a due domande:

- Di quanto posso aumentare/diminuire il profitto senza che mi cambi il piano di produzione, rimanendo ottimo;
- Per fare un'analisi sulla stabilità del modello: ho una soluzione ottima, ma quanto questa è stabile rispetto al coefficiente della funzione obiettivo? Cioè qual è il range di variazione del coefficiente della funzione obiettivo che mi garantisce che la soluzione mi rimane ottima? Per esempio, se ci troviamo in una situazione in cui le soluzioni sono multiple (due vertici e come possibili soluzioni tutte quelle comprese fra questi due vertici), la soluzione è *instabile*: ho due soluzioni ottime di base e infinite soluzioni ottime; ma se vario di pochissimo uno dei due coefficienti la mia soluzione si sposterà di molto dalla situazione attuale!

Quindi il report dei risolutori si possono ottenere le seguenti informazioni:

- Di quanto posso cambiare i coefficienti della funzione obiettivo senza che cambi la soluzione ottima, cioè senza che cambi la mia decisione;
- L'impatto sulla soluzione ottima, ovvero sulla funzione obiettivo del cambiamento dei termini noti dei vincoli, oppure dei coefficienti delle variabili;

Se cambiamo i termini noti il vincolo si sposta e cambia la mia soluzione e aumenta o diminuisce la mia funzione obiettivo a seconda che si allarghi la regione ammissibile o la si diminuisca. Se si sposta un vincolo e lo si rende più facile/meno facile da soddisfare (quindi si aumenta o diminuisce il valore RHS), il tasso di cambiamento della funzione obiettivo al variare del termine noto è **costante**, perché il punto si sposta lungo una retta, ovvero:

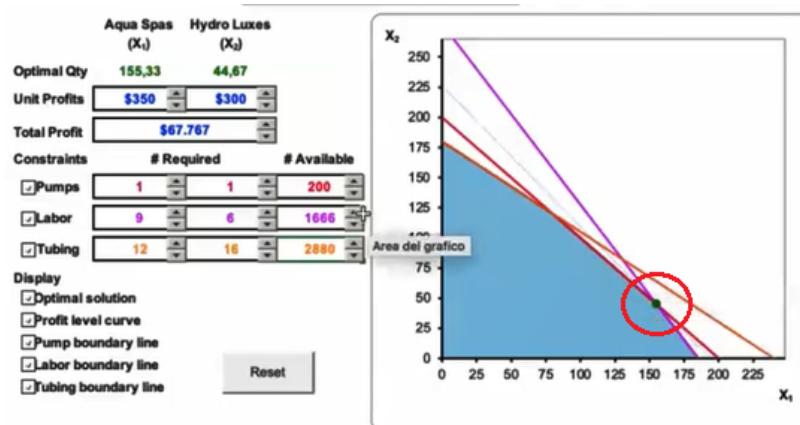


Figure 8.5: Prima

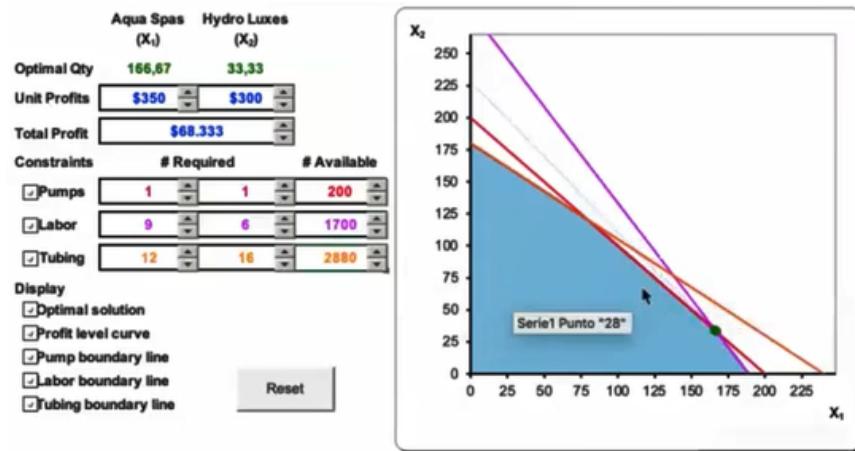


Figure 8.6: Dopo

Il punto si sposta sulla retta rossa. Il tasso di cambiamento della funzione obiettivo al variare del termine noto, lo ripetiamo, è costante perché il punto si sposta lungo una retta. Continuando ad aumentare o diminuire si arriva a cambiare倾inazione e non ci si sposta più lungo la retta rossa ma lungo quella arancione. Il tasso di variazione della funzione obiettivo al variare del mio termine noto **cambia**. Quello che i risolutori mi forniscono è il **tasso di variazione della funzione obiettivo al variare del termine noto**, insieme ad un range di variazione in cui questo tasso rimane valido perché questo tasso rimane valido per un certo range di variazione e poi cambia. Quindi nel risolutore si può trovare questa informazione rispondendo alla domanda *Cosa succede se cambio il valore del termine noto? La funzione obiettivo varierà di un certo tasso*. E questo tasso avrà valenza in un certo range. Quindi mi dà i termini di validità di questo tasso.

L'analisi di sensitività comprende anche il valore delle variabili slack: se sono uguali a 0 il vincolo è attivo e dalla soluzione del nostro problema possiamo anche vedere quali sono le risorse che sono critiche, che sono quelle che hanno una corrispondente variabile slack uguale a 0, cioè sono quelle che determinano i vincoli all'intersezione dei quali ho la mia funzione obiettivo. Significa che sono quei vincoli che stanno vincolando la mia soluzione da migliorare.

Rivedendo cosa succede quando si cambia il coefficiente della funzione obiettivo:

How Changes in Objective Coefficients Change the Slope of the Level Curve

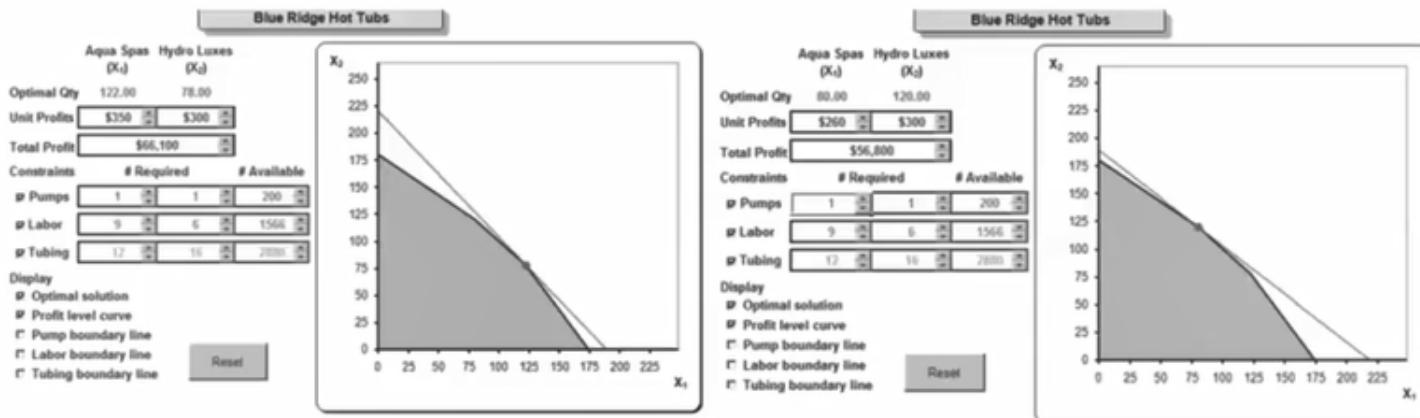


Figure 8.7: Cambio funzione obiettivo

Un esempio di report del risolutore:

Sensitivity Report

```
printSensitivityObj(model)
## Objs Sensitivity
## 1 C1 300 <= C1 <= 450
## 2 C2 233.333333333333 <= C2 <= 350
```

The objective coefficient C_1 was 350 therefore it can increase up to \$100 or decrease up to \$50 without changing the optimal solution assuming all other coefficients remain constant

Similarly, the objective function value C_2 can increase by \$50 or decrease by \$66.67 without changing the optimal values of the decision variables, assuming all other coefficients remain constant.

Figure 8.8: Cambio funzione obiettivo

Questo significa che in questo problema il coefficiente C_1 può cambiare fra 300 e 450 senza che cambi il valore della soluzione ottimale. Quindi per valori compresi tra questi due potrei ottenere soluzioni multiple. Al di fuori, potrebbe cambiare la soluzione ottima, ma se cambio C_1 all'interno di questi valori la soluzione ottima non cambia. Lo stesso ragionamento per C_2 .

ATTENZIONE: questo vale se si cambia **un solo coefficiente alla volta**. Se cambio entrambi i coefficienti, ciò potrebbe non essere più valido.

Tali range sono definiti **range di ottimalità** e valgono assumendo che tutti gli altri coefficienti rimangano fissi.

ATTENZIONE: quando i coefficienti della funzione obiettivo variano allora la regione ammissibile rimane invariata.

Cosa succede se modifico i **termini noti**? Abbiamo visto che cambiando il termine noto di un vincolo **cambia** la regione ammissibile, quindi il punto di ottimo si sposta. Anche in questo caso abbiamo detto che abbiamo il valore del termine noto che nel nostro problema era 200 e 1566 e 2880; i primi due vincoli sono attivi (il loro valore finale è uguale al RHS) e un terzo no.

Cosa vi possiamo vedere? Il **shadow price** è il **tasso di variazione della funzione obiettivo** al variare del corrispondente termine noto. Se un vincolo non è attivo, quindi il punto di ottimo non risiede su quel vincolo, se anche sposto quel vincolo, non succede nulla alla mia soluzione perché rimane lì. Il tasso di variazione della funzione obiettivo al variare del termine noto sarà 0. Quindi, per i vincoli non attivi il tasso di variazione della funzione obiettivo al variare del termine noto è pari a 0. Per i vincoli attivi, se cambio il termine noto, si sposta il corrispondente punto di soluzione. Questo tasso rimane costante entro un certo range di variazione che anche qua possiamo leggere dai report:

Changes in Constraint RHS Values

- The **shadow price** of a constraint indicates the amount by which the objective function value changes given a unit *increase* in the RHS value of the constraint, *assuming all other coefficients remain constant*.
- Shadow prices hold only within RHS changes falling within a range
- Shadow prices for nonbinding constraints are always zero.

Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
Pumps Req'd Used	200	200	200	7	26
Labor Req'd Used	1566	17	1566	234	126
Tubing Req'd Used	2712	0	2880	1E+30	168

Figure 8.9: Shadow price

Un esempio:

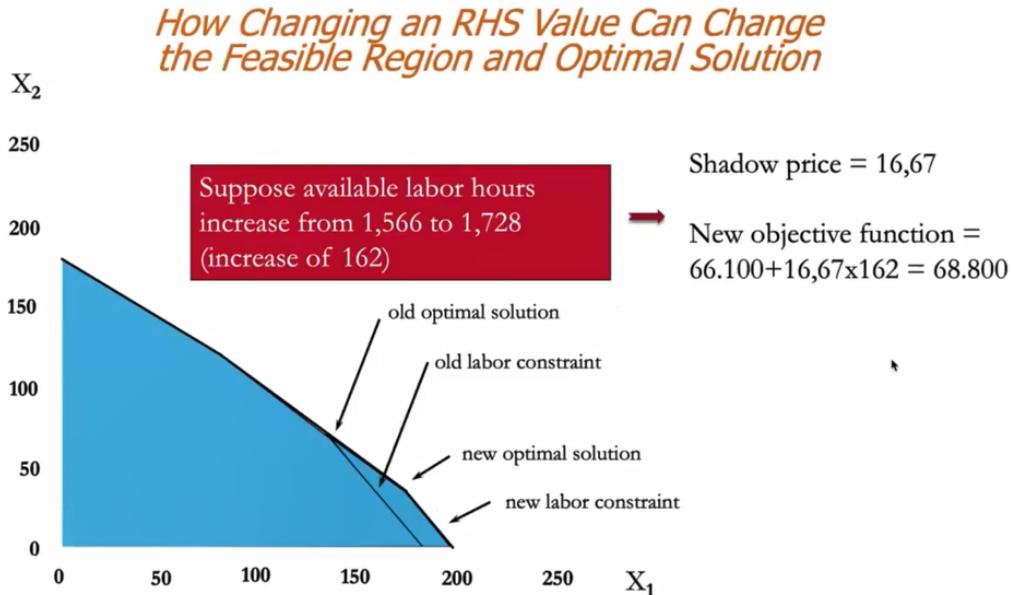


Figure 8.10: Shadow price

Quindi i **prezzi ombra** indicano il cambiamento che avremo nella funzione obiettivo al variare del termine noto. Se vario il termine noto di un vincolo attivo allora cambia la regione ammissibile e la regione corrispondente. Quello che mi fornisce il report è il tasso di variazione al variare della funzione obiettivo; quello che non mi dà il report è il valore della mia decisione. Per trovare il valore della nuova soluzione devo rifare dei calcoli: ho la base, quindi i calcoli si riducono alla risoluzione di un sistema di equazioni con il nuovo coefficiente.

Chapter 9

LEZIONE 7 - Sensitivity ex. 6-7

Esercizio improntato alla programmazione lineare e vedremo concetti relativi all'analisi di sensitività.

Piccola introduzione: una volta risolto un problema, soprattutto in molti ambiti relativi all'ingegneria, è importante chiedersi quanto la soluzione ottima cambi, se cambiano i dati in input. Questo perché la soluzione ottima potrebbe essere instabile, ovvero difficile da raggiungere. Nella realtà, la definizione dei coefficienti che definiscono il modello è comunque soggetta a errori di vario tipo, errori di stima o approssimazione. Noi vedremo che all'interno della PL esistono degli strumenti metodologici che ci permettono di analizzare cosa succede quando cambiano i coefficienti della funzione obiettivo (**i coefficienti dei costi, i coefficienti di profitto** a seconda del problema), oppure cosa succede quando cambiano i valori dei termini noti (**il right-hand side**).

Cosa potrebbe succedere?

- **Primo caso:** le variabili di base che definiscono quali variabili decisionali sono attive e quindi anche quali vincoli sono attivi rimangono invariate. Il vertice del politopo non cambia. Questo non significa che la sua posizione non cambi. Significa solo che a seguito della trasformazione anche di questo politopo (se modifichiamo il vettore dei termini noti lo stiamo deformando), il vertice non viene a cambiare. In questo caso possiamo ricalcolarne la nuova posizione e il nuovo valore ottimo;
- **Secondo caso:** il vertice cambia, il politopo si trasforma in tale maniera da far cambiare il vertice della posizione ottima. Non è solo questo il caso: immaginiamo se dovessimo tornare agli esercizi precedenti, la scelta di un determinato valore di profitto portava all'attivazione o meno di una certa variabile. Ciò equivale alla scelta di un vertice, ovvero ad un cambio di pendenza della linea di livello che rappresenta la nostra funzione obiettivo che quindi intercetta il politopo (il dominio) in un punto diverso. In questo caso abbiamo ben poco da fare se non ridefinire la nuova base per il calcolo del nuovo vertice; da questa calcolare i nuovi valori delle nostre variabili decisionali e infine, con questi valori, ottenere il valore ottimo della funzione obiettivo.

Cerchiamo di analizzare tali due aspetti separatamente (ovvero la variazione dei coefficienti della funzione obiettivo e una piccola variazione del vettore dei termini noti). Consideriamo il seguente problema (già visto precedentemente):

$$\begin{aligned} & \text{MAX : } 20X_1 + 60X_2 \\ & \text{S.t. : } 30X_1 + 20X_2 \leq 2700 \\ & \quad 5X_1 + 10X_2 \leq 850 \\ & \quad X_1 + X_2 \geq 95 \\ & \quad X_1, X_2 \geq 0; \end{aligned} \tag{9.1}$$

Si tratta di un problema di ottimizzazione della produzione in cui avevamo dei vincoli di capacità di budget e un vincolo di bootstrap che ci diceva che la somma di quanto produciamo doveva essere per forza di cosa maggiore/uguale a 95. Avevamo trovato una soluzione iniziale e ora ci chiediamo quanto cambia questa soluzione quando modifichiamo in maniera ridotta, per piccole modifiche, di uno

dei coefficienti della funzione obiettivo. Alteriamo la funzione obiettivo e cambiamo il primo dei due coefficienti, portandolo da 20 a 25.

Let's consider a **small** alteration in the model, for instance suppose that the profit coefficient for Product 1 changes **from 20 to 25**.

From the figure below we can see that the optimal basis does not change and the solution (the vertex) remains the same. In fact, both $f(x_1, x_2) = 20x_1 + 60x_2$ and $\hat{f}(x_1, x_2) = 25x_1 + 60x_2$ are maximized in $x_1^* = 20$, $x_2^* = 75$, $s_1 = 600$ (Basic variables $[x_1, x_2, s_1]$).

Since the solution does not change then the improvement in the optimal value is easy to compute: $\Delta f = 20 \times (25 - 20) = 100$. The value of the objective function $\hat{f}(x_1, x_2)$ is thus $4900 + \Delta f = 5000$.

Let's consider now a *more drastic* change in the profit of Product 1, for example imagine that the profit increases **from 20 to 30**. We can see in the figure that the optimal basis changes and identifies a different vertex in which the basic variables are $x_1 = 50$, $x_2 = 60$, $s_3 = 15$. The value of the new objective function (5600) is now not directly computable from the original value.

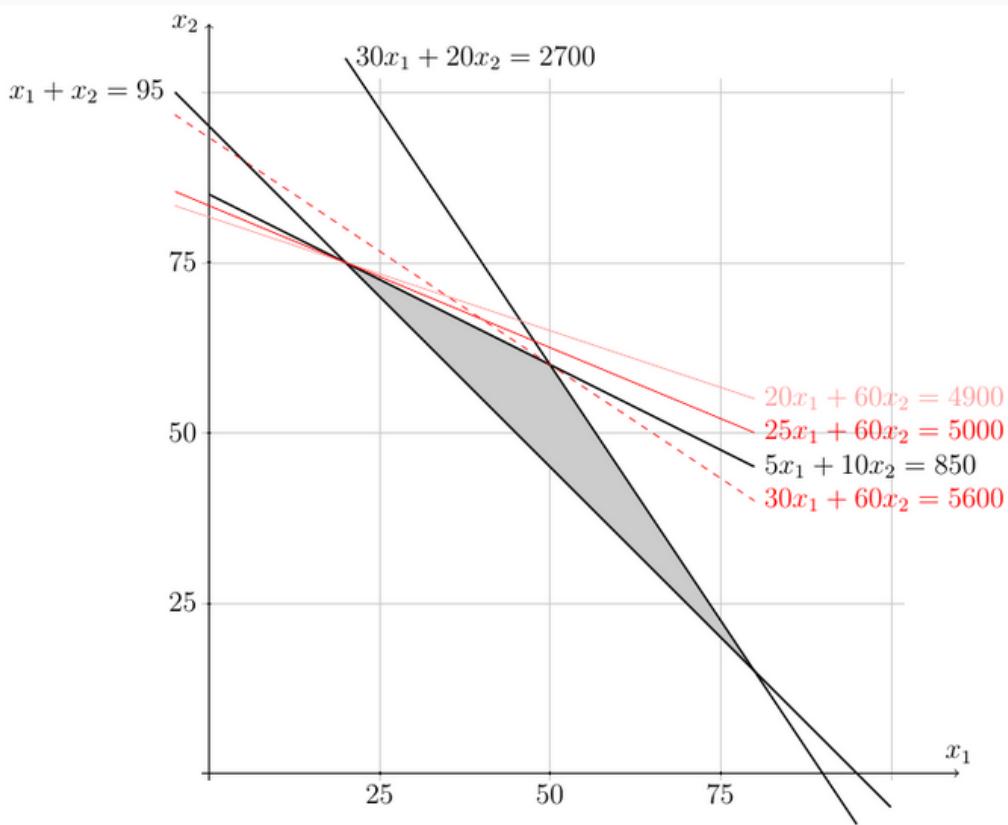


image2

Figure 9.1: Changing coefficients

9.0.1 Primo scenario

In grigio troviamo il dominio, l'area definita dai tre vincoli; in rosso più chiaro la curva di livello relativa alla soluzione ottima del problema iniziale (non alterato) e avevamo individuato il punto ottimo nella sezione in alto a sinistra che aveva $X_1 = 20$ e $X_2 = 75$ e il valore ottimo della funzione obiettivo per questo problema era 4900. Cosa succede se consideriamo la funzione obiettivo che segue?

$$25X_1 + 60X_2 \quad (9.2)$$

L'inclinazione delle curve di livello cambia un poco, ma non tanto da cambiare la posizione del punto ottimo, perché continueremo a disegnare curve di livello fra loro parallele fino a individuare quella che è l'ultima curva di livello che per ultima tocca la nostra area feasible, cioè quella che contiene soluzioni per

noi accettabili. Quindi per noi il vertice rimane invariato (sia nella **base** costituita dalle variabili X1 e X2 e la variabile di slack S1) anche se la soluzione della funzione obiettivo cambia. Questo significa che per la variazione di questo coefficiente di 5 unità la soluzione ottima non cambia. Ma cambierà il valore della funzione obiettivo! Perché abbiamo aumentato di 5 unità il valore del coefficiente di profitto. Quindi per ottenere il nuovo valore ottimo, noi possiamo considerare il vecchio valore ottimo più l'incremento nel guadagno - un Δf (o il decremento dei costi a seconda del modello) pari alla variazione del nostro coefficiente che è di 5 unità per il valore assunto dalla variabile decisionale X1. Il nuovo valore della funzione obiettivo assunto nel punto che massimizza il nostro profitto sarà pari a 5000.

Dimostriamo che quanto detto sia vero solo nell'intorno dei coefficienti originali. Se il nostro valore X1 assume valore 30 succede che (la curva tratteggiata), succede che l'inclinazione della curva è tale per cui l'ultimo punto del politopo intercettato da questa curva è il punto che si trova a (50, 60) nel primo quadrante. Questo vertice corrisponde ad un'altra soluzione di **base**, ma al posto di S1, variabile di slack attiva nel primo caso, si avrà $S3 = 15$. Non solo quindi cambierà il valore della funzione obiettivo, ma cambierà proprio il punto ottimo, ovvero **il vertice che verrà selezionato**. Dovremo ricalcolare la posizione di questo vertice e ottenuto questa posizione otterremo il valore di questa funzione obiettivo pari a 5600.

9.0.2 Secondo scenario

/// Variation of the right-hand side of the constraint set

Imagine that in our example the number of machine hours increases from 850 to 900 hours (constraint 2). Let's see how the feasible region changes:

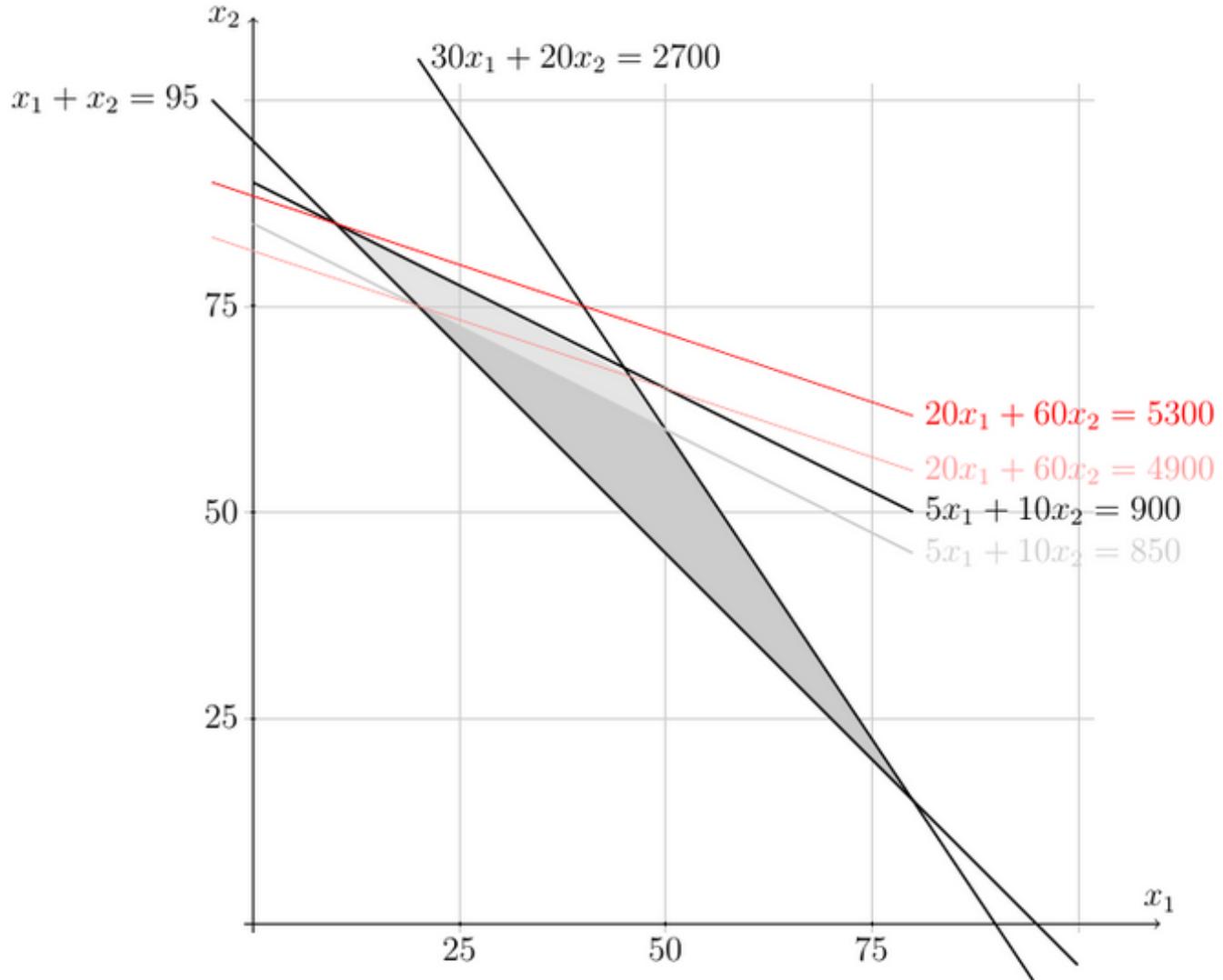


image3

Figure 9.2: Secondo scenario

Supponiamo che il termine noto di uno dei nostri vincoli non sia stato stimato correttamente. Per esempio, immaginiamo che il termine noto associato al numero di ore macchina disponibili, quindi come vincolo di budget, sia spostato da 850 a 900 ore, perché la nostra stima iniziale era sbagliata. (ci stiamo riferendo al vincolo $5X_1 + 10X_2 \leq 850$. Il nuovo vincolo diventerà $5X_1 + 10X_2 \leq 900$).

In questo caso vediamo come cambia il nostro problema: siccome stiamo lavorando sull'insieme dei vincoli quello che otterremo è una modifica, un cambiamento nella forma dello spazio delle soluzioni ammissibili. Stiamo spostando più in alto rispetto al precedente il vincolo che definisce la parte superiore dello spazio delle soluzioni ammissibili. Abbiamo ingrandito l'area e avremo ottenuto un nuovo politopo. Se ora l'area è più grande, ovvero il numero delle soluzioni potenzialmente ammissibili è maggiore, la soluzione ottima di questo problema avrà un valore maggiore rispetto a quella del problema originale. Questo perché abbiamo rilassato un vincolo e avremo quindi più risorse. Potrò massimizzare di più il mio

guadagno perché potrò usare risorse in più. In generale, se estendo lo spazio delle soluzioni ammissibili mi trovo ad avere più soluzioni e quindi potenzialmente ho la possibilità di ottimizzare di più: il che significa avere maggiore valore della funzione obiettivo. Ma succede sempre? No, soltanto quando stiamo espandendo e stiamo avendo effetto **sul vertice che inizialmente era ottimo**. Se avessimo allargato lo spazio delle soluzioni ammissibili tramite il rilassamento del vincolo $30X_1 + 20X_2 = 2700$, non avremmo ottenuto un miglioramento nella funzione che vogliamo ottimizzare.

Abbiamo quindi un nuovo vincolo:

$$5x_1 + 10x_2 = 900 \quad (9.3)$$

La funzione obiettivo non è cambiata, ma il vertice corrispondente all'intercettarsi dei due vincoli $x_1 + x_2 = 95$ e $5x_1 + 10x_2 = 900$ si è spostato in alto. La nostra curva di livello intercetterà questo punto in un'altra posizione e il valore della funzione obiettivo cambierà. La posizione del vertice è cambiata, ma la sua definizione in termini di quali sono i vincoli che si devono attivare per poter definire quel vertice **non sono cambiati**. Questo equivale a dire che la sua **base non è cambiata** e sarà ancora x_1, x_2, S_1 e questa corrisponde alla matrice di rango massimo estratta dalla matrice dei coefficienti: (con l'aggiunta del valore unitario associato ad S_1)

$$\begin{pmatrix} 30 & 20 & 1 \\ 5 & 10 & 0 \\ -1 & -1 & 0 \end{pmatrix}$$

per il vettore dei termini noti:

$$\begin{pmatrix} 2700 \\ 900 \\ -95 \end{pmatrix}$$

e moltiplicando l'inversa della prima matrice con la seconda si otterrà quanto seguono:

We notice that the basis (x_1, x_2, s_1) remains optimal (scenario a.) but the position of the vertex identified has changed. The new solution is:

$$\begin{pmatrix} x_1 \\ x_2 \\ s_1 \end{pmatrix} = \begin{bmatrix} 30 & 20 & 1 \\ 5 & 10 & 0 \\ -1 & -1 & 0 \end{bmatrix}^{-1} \begin{pmatrix} 2700 \\ 900 \\ -95 \end{pmatrix} = \begin{pmatrix} 10 \\ 85 \\ 700 \end{pmatrix} \quad s_2 = s_3 = 0$$

Figure 9.3: Nuova soluzione

la cui nuova posizione sarà $(10, 85)$ ed $S_1 = 700$. Quindi la base non è cambiata perché la matrice estratta è la stessa, ma cambia la posizione del vertice che si è spostata da $20, 75$ a $10, 85$ e possiamo sfruttare questa informazione per calcolare l'impatto di questa variazione sulla funzione obiettivo, ovvero:

$$c_B^T B^{-1} \Delta b = (20 \ 60 \ 0) \begin{bmatrix} 30 & 20 & 1 \\ 5 & 10 & 0 \\ -1 & -1 & 0 \end{bmatrix}^{-1} \begin{pmatrix} 0 \\ 50 \\ 0 \end{pmatrix} = 400$$

where c_B^T denotes the cost/profit of the basic variables and Δb is a vector containing only the difference in the values of the RHS.

Figure 9.4: Delta di cambio

Il nuovo valore sarà dato da 20×-10 più 60×10 . Questo ci permette di capire che l'incremento, il miglioramento della nostra funzione obiettivo, il nostro delta di f che abbiamo calcolato prima è pari a 400. Quindi se il valore originale era 4900 il nuovo valore della funzione obiettivo sarà 5300.

Si può calcolare tale variazione senza ricalcolare la posizione del nuovo vertice? Lo si può fare essenzialmente considerando non il vettore B dei termini noti interamente in tale moltiplicazione, ma soltanto il **delta B** , cioè la variazione nei termini noti, che conosciamo perché l'abbiamo imposta noi che sarà pari a $(0 \ 50 \ 0)$ (per il secondo termine noto abbiamo una variazione di 50 unità). (CB trasposto è il vettore dei coefficienti della funzione obiettivo associate alle variabili di base x_1, x_2, S_1). Lo zero che vediamo viene associato alla prima variabile di slack. Essenzialmente quando aggiungiamo le variabili di slack stiamo implicitamente dicendo che quelle variabili non hanno impatto sulla funzione obiettivo e quindi sono associate a coefficienti nulli. *Delta B* contiene solo la differenza dei valori dei vettori dei termini noti. E il prodotto:

$$y^T = c_B^T B^{-1} \quad (9.4)$$

vengono chiamate **variabili duali del problema**: rappresentano il valore delle variabili di un problema duale, costruito a partire dal problema considerato facendo diventare i vincoli delle variabili e le variabili dei vincoli. Pertanto siccome ho m vincoli (in questo caso 3) le variabili duali assumeranno tre valori. Vengono anche chiamate **prezzi ombra - shadow prices** perché quantificano quanto si guadagna o si perde aumentando o diminuendo di una unità il valore di una certa risorsa. Soltanto analizzando il valore di queste variabili duali, posso dire quanto guadagno se riesco ad aggiungere una unità di questa risorsa, ovvero se aumento di una unità il termine noto di uno dei vincoli.

La sensibilità associata alla soluzione ottima a fronte di un cambio nel termine noto di uno dei coefficienti del termine noto ha una **valenza locale**: ovvero, aumentando di molto oltre una certa soglia il valore di un termine noto si modifica la regione ammissibile in maniera tale da cambiare la definizione del punto ottimo, ovvero del vertice.

Infatti, guardando il seguente grafico:

Nonetheless, shadow prices have only a **local validity**; indeed, if the number of hours suffers a large change, let say that b_2 rises to 1000 hours, then the current basis (x_1, x_2, s_1) will not be optimal, as you can see in the following picture:

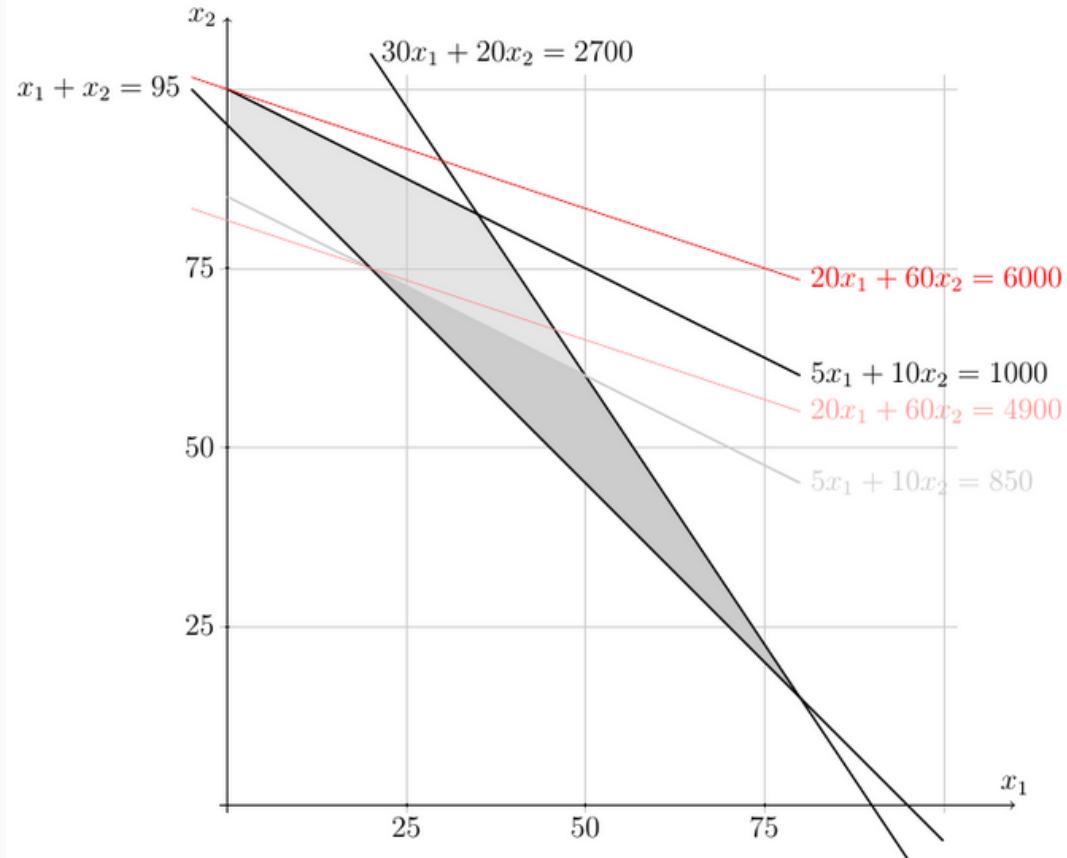


image4

Figure 9.5: Validità locale degli shadow price

Ad esempio, consideriamo di estendere il termine noto del nostro secondo vincolo da 850 a non più 900 ma a 1000. Quello che vediamo nel grafico precedente è come cambia l'area dello spazio ammissibile considerando questo nuovo vincolo. L'inclinazione della curva di livello associata alla funzione obiettivo non è cambiata, quindi il nostro vertice ottimo sarà sempre nella parte alta dello spazio delle soluzioni ammissibili; ma siccome ora deve valere il vincolo che X_1 sia maggiore di 0, il vertice non è più definito totalmente dall'intersezione tra il vincolo 2 e il vincolo 3, ma è definito dall'intersezione tra il vincolo 2 (questo nuovo vincolo con 1000 come termine noto) e il bound $x_1 \geq 0$, quindi il vertice in alto a sinistra. Inoltre, non sarà più un triangolo, il numero di vertici del nostro politopo è salito a 4 modificando sostanzialmente la geometria del nostro politopo. Quindi è cambiata anche la base del nostro vertice che non conterrà più X_1 ma soltanto X_2 e le due variabili di slack che sono S_1 e S_3 . A questa soluzione è associata una nuova matrice B di base che contiene il vettore della matrice dei coefficienti associato ad X_2 che è $(20 \ 10 \ -1)$ e poi abbiamo due vettori unitari associati alla variabile di slack S_1 ed S_3 che è la terza posizione che è un 1. Si ha poi il vettore dei termini noti e si ottiene un nuovo punto che avrà valore $(100 \ 700 \ 5)$, dove 100 è il valore della variabile X_2 mentre X_1 varrà 0, in quanto non è più una variabile di base. (tutte le variabili non di base assumeranno valore 0). Con questi due valori potremo quindi calcolare il valore della nuova funzione obiettivo che sarà appunto incrementato a 6000.

Now the new optimal basis is $\{x_2, s_1, s_3\}$ and the solution is

$$\begin{pmatrix} x_2 \\ s_1 \\ s_3 \end{pmatrix} = \begin{bmatrix} 20 & 1 & 0 \\ 10 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}^{-1} \begin{pmatrix} 2700 \\ 1000 \\ -95 \end{pmatrix} = \begin{pmatrix} 100 \\ 700 \\ 5 \end{pmatrix} \quad x_1 = s_2 = 0$$

with a $\max f(x_1, x_2) = 6000$.

Figure 9.6: Nuova soluzione

Risoluzione del problema con LPSolve tramite R - Ex.6 e Ex. 7.

Chapter 10

LEZIONE 8 - INTEGER PROGRAMMING

Iniziamo a guardare i problemi di programmazione lineare intera: i valori delle variabili sono limitati ad assumere valori interi. I problemi di ottimizzazione intera hanno molteplici applicazioni: in molti problemi decisionali devo decidere una quantità intera che non può essere continua. Nel caso di Acqua-Spa e delle Hot tubs, il numero dei prodotti da mettere in produzione sarà un numero intero. Quando risolviamo un problema di PL senza considerare i vincoli di interezza delle variabili si può ottenere una soluzione nella quale alcuni dei valori possono essere continui e non interi. Oppure si possono avere i problemi di **scheduling**, i problemi di **manufacturing** (se devo costruire aeroplani, ne posso costruire solo interi); tutti problemi in cui devo considerare delle soluzioni logiche come aprire/non aprire uno stabilimento; percorrere/non percorrere un certo arco di un grafo.

Se il problema presume l'esistenza di variabili intere deve aver aggiunto questo vincolo. E' anche possibile che solo alcune delle variabili siano intere: in questo caso avremmo un problema di PL misto.

Relaxation

- Original ILP

$$\begin{aligned} \text{MAX: } & 2X_1 + 3X_2 \\ \text{S.T.: } & X_1 + 3X_2 \leq 8.25 \\ & 2.5X_1 + X_2 \leq 8.75 \\ & X_1, X_2 \geq 0 \\ & X_1, X_2 \text{ must be integers} \end{aligned}$$

- LP Relaxation

$$\begin{aligned} \text{MAX: } & 2X_1 + 3X_2 \\ \text{S.T.: } & X_1 + 3X_2 \leq 8.25 \\ & 2.5X_1 + X_2 \leq 8.75 \\ & X_1, X_2 \geq 0 \end{aligned}$$

Figure 10.1: PL intera

Se risolviamo il problema di PL senza tenere in conto i vincoli di interezza abbiamo risolto la sua versione rilassata. Il problema di PL rilassato associato ad un problema di PL intera è il problema stesso senza i vincoli di interezza. Perché ne parliamo? In realtà non abbiamo un metodo analogo al simplex per risolvere problemi di PL intera. Il simplex si basava sul fatto che le soluzioni ottime si trovavano sui vertici della regione ammissibile e purtroppo qui la regione ammissibile definita dai vincoli, esclusi quelli di interezza, è una regione che contiene tutti i punti interi e non, e i suoi vertici non è garantito siano in punti in cui le variabili sono intere. Infatti, abbiamo una situazione di questo tipo:

Integer Feasible vs. LP Feasible Region

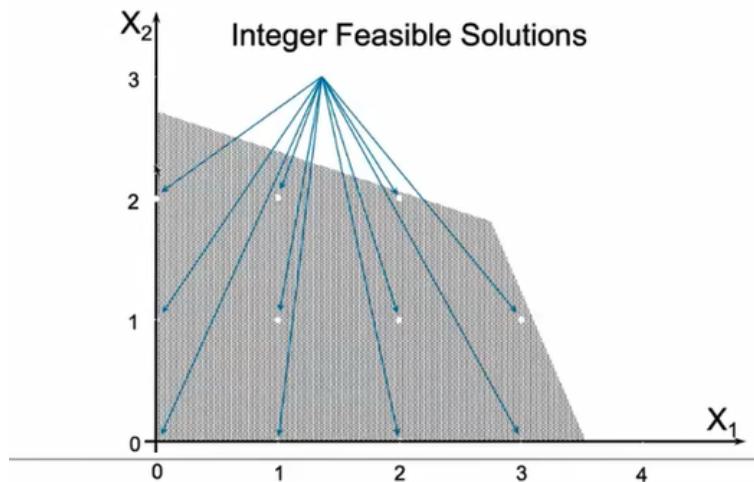


Figure 10.2: PL intera

Quindi, abbiamo dei vincoli funzionali che ci definiscono la regione ammissibile, ma all'interno di questa regione ammissibile **solo i punti interi** sono effettivamente ammissibili per il nostro problema di PL intera. E quindi è vero che se la regione è finita, i punti li possiamo in qualche modo contare più facilmente, ma nelle applicazioni reali, valutare tutti questi interi sarebbe veramente computazionalmente oneroso. Dobbiamo trovare una strategia che ci aiuti a trovare le soluzioni ottime intere.

La prima cosa che viene in mente è quella di risolvere il problema rilassato, quindi su tutta la regione, senza considerare i vincoli di interezza, per il quale abbiamo un algoritmo come il simplex che è un algoritmo efficiente, e poi arrotondare alla soluzione più vicina. Purtroppo, **non funziona tale approccio**.

Integrality Conditions

$$\begin{aligned}
 & \text{MAX: } 350X_1 + 300X_2 && \} \text{ profit} \\
 & \text{S.T.: } 1X_1 + 1X_2 \leq 200 && \} \text{ pumps} \\
 & & 9X_1 + 6X_2 \leq 1566 & \} \text{ labor} \\
 & & 12X_1 + 16X_2 \leq 2880 & \} \text{ tubing} \\
 & X_1, X_2 \geq 0 && \} \text{ nonnegativity} \\
 & X_1, X_2 \text{ must be integers} && \} \text{ integrality}
 \end{aligned}$$

Integrality conditions are easy to state but make the problem much more difficult (and sometimes impossible) to solve.

Figure 10.3: Integrality condition

Aggiungendo le condizioni di integralità rendono il problema molto più difficile e alcune volte impossibile da risolvere in tempi ragionevoli. Quindi, cosa significa risolvere un problema di PL intera?

- Se succede che risolvendo un problema di programmazione rilassata otteniamo una soluzione intera, diciamo che siamo stati fortunati e possiamo considerare quella come soluzione ottima: questo era

il caso del *Blue Ridge Hot Tubs*. Ma cosa succede ad esempio se riduciamo il numero di ore di lavoro a 1520 e la quantità di tubazioni a 2650?

Integrality Conditions

$$\begin{aligned}
 & \text{MAX: } 350X_1 + 300X_2 && \} \text{ profit} \\
 & \text{S.T.: } 1X_1 + 1X_2 \leq 200 && \} \text{ pumps} \\
 & & 9X_1 + 6X_2 \leq 1520 & \} \text{ labor} \\
 & & 12X_1 + 16X_2 \leq 2650 & \} \text{ tubing} \\
 & & X_1, X_2 \geq 0 & \} \text{ nonnegativity} \\
 & & X_1, X_2 \text{ must be integers} & \} \text{ integrality}
 \end{aligned}$$

Optimal solution of the relaxed problem:

$$X_1 = 116,9444 \quad X_2 = 77,9167$$

Corresponding to a maximum profit of \$64306

Figure 10.4: Optimal solution

Ottenendo una soluzione che è continua e che non è un numero intero. Che cosa viene in mente di fare?

- Rounding: la prima cosa è quella di sostituire il valore non intero con un suo arrotondamento. Questo, in generale, non funziona in quanto:
 - la soluzione arrotondata potrebbe essere innanzitutto non ammissibile;
 - nel caso in cui lo fosse, potrebbe essere non ottimale.

Ad esempio, arrotondando all'intero più vicino si avrà che:

How Rounding Down Can Result in an Infeasible Solution

Blue Ridge Hot Tubs			
	Aqua-Spas	Hydro-Luxes	Total Profit
Number to Make	117	78	
Unit Profits	\$350	\$300	\$64,350
Constraints			
Pumps Req'd	1	1	Used 195 Available 200
Labor Req'd	9	6	Used 1521 Available 1520
Tubing Req'd	12	16	Used 2652 Available 2650

Figure 10.5: Rounding problem

Così facendo notiamo che:

- il primo vincolo è soddisfatto, infatti $195 < 200$;
- il secondo e il terzo vincolo non sono soddisfatti, infatti sto utilizzando più risorse di quelle disponibili e la **soluzione non è ammissibile**.

Quindi l'arrotondamento ad un intero più vicino, potrebbe ritornarci una soluzione non ammissibile. Infatti, ad esempio:

How Rounding Down Can Result in an Infeasible Solution

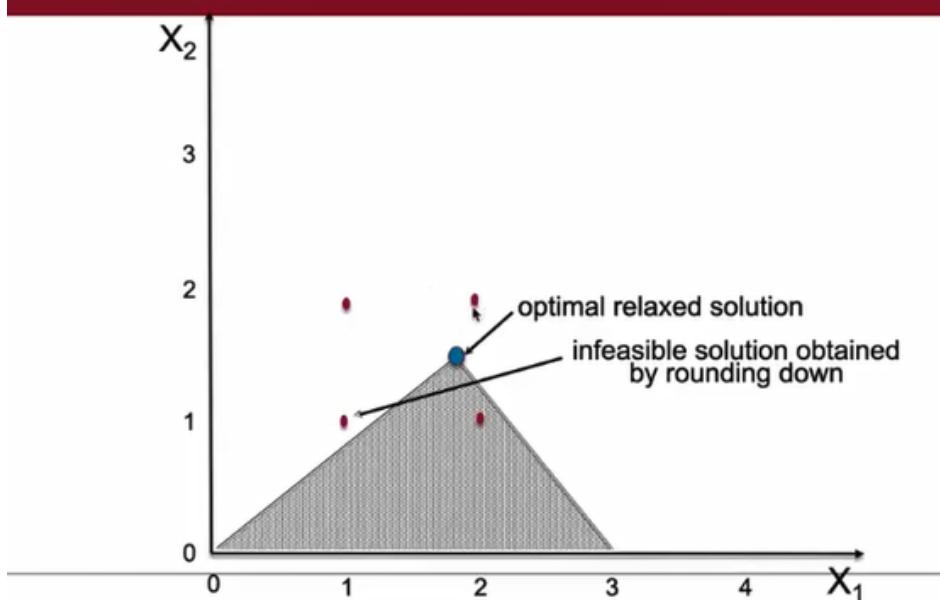


Figure 10.6: Problem solution

Arrotondando all'intero più vicino, per eccesso o per difetto, possiamo ottenere delle soluzioni non ammissibili (3 soluzioni su 4 non ammissibili). Quindi la soluzione non è facilmente ottenibile arrotondando. Anche se fra tutti gli arrotondamenti possibili trovasse una soluzione ammissibili, non avrei la certezza che questa sia effettivamente la soluzione ottima. In ogni caso l'arrotondamento e la successiva soluzione che potrebbe essere ammissibile potrebbe tornare utile come guida, un buon punto di partenza, anche se non ci garantisce l'ottimalità della soluzione.

Esempio di arrotondamento per difetto:

How Rounding Down Can Result in an Infeasible Solution

Blue Ridge Hot Tubs			
	Aqua-Spas	Hydro-Luxes	Total Profit
Number to Make	116	77	
Unit Profits	\$350	\$300	\$63,700
Constraints			
Pumps Req'd	1	1	193 200
Labor Req'd	9	6	1506 1520
Tubing Req'd	12	16	2624 2650

Figure 10.7: Arrotondamento per difetto

Tutti e tre i vincoli sono soddisfatti, infatti non utilizzo più risorse del necessario, ma ha un profitto più basso di quello che potremmo ottenere.

10.0.1 Algoritmo di Branch-and-Bound

Tale algoritmo può essere utilizzato in questi casi per risolvere un problema di PL intera. Esistono altre soluzioni, ma noi vedremo questa in particolare. Quello che succede con il B&B è che richiede la soluzione di una serie di problemi di PL: riduce il problema di PL intera ad una serie di problemi di PL lineare per i quali possiamo usare l'algoritmo del simplex. Tale procedura che risolve sequenzialmente diversi problemi di PL, teoricamente ci garantisce di trovare la soluzione, se esiste, di qualsiasi problema di PL intera in quanto si ha la convergenza.

In pratica è computazionalmente oneroso: lo sforzo computazionale necessario per risolvere all'ottimalità un problema di PL intera con il metodo del B&B può essere inaccettabile in termini di tempo. Vedremo quali strategie adottare per valutare la soluzione fra quelli ottenuti ammissibili dall'algoritmo. Ciò che è costoso non è tanto ottenere delle soluzioni ammissibili quanto piuttosto dimostrare che fra queste effettivamente ci sia quella ottima.

Ciò che ci verrà in soccorso sarà la capacità di valutare la qualità della soluzione fin qui ottenuta. E quindi potremo valutare la qualità della soluzione e se essa è soddisfacente per il nostro problema ci si può fermare. I pacchetti software che implementano il B&B chiedono di specificare un **fattore di tolleranza** che è quello che deciderà di fermare il B&B quando si raggiunge una soluzione ammissibile intera che è all'interno del range di tolleranza. Questo permette di interrompere la procedura quando la soluzione ottenuta è vicina alla soluzione ottima, in termini percentuali. In questo caso quindi, i bounds ottenuti dal rilassamento lineare del problema possono esserci utili. Perché? Perché quando ottengo la soluzione rilassata del problema (quindi togliendo i vincoli di interezza) si ottiene una soluzione che sarà il miglior valore, la soluzione ottima del problema rilassato (quindi la soluzione migliore che posso ottenere in termini di funzione obiettivo rispetto a qualsiasi soluzione intera - se sto massimizzando sarà un upper bound, altrimenti un lower bound per la soluzione del problema di PL Intera).

La soluzione del problema rilassato può darmi qualche indicazione rispetto alla qualità delle soluzioni ammissibili intere che via via posso ottenere durante la procedura del B&B. Quindi, per esempio:

- Il valore della soluzione del problema rilassato era 64306 dollari;
- Supponendo di accettare una soluzione ottima dentro un range del 95% di questa soluzione ottima: possiamo ovvero accettare qualsiasi soluzione lineare intera che sia maggiore/uguale di 61090 dollari;
- Se il valore della funzione obiettivo di una soluzione ammissibile intera è uguale a 61090 o maggiore, allora starà all'interno del 5% della soluzione ottimale, ovvero disterà meno del 5% della soluzione ottima che si potrà ottenere. Ovviamente questa è una stima dell'upper bound in quanto la soluzione ottima intera non sarà mai 64306.

Quindi la soluzione ottima del problema rilassato ci è utile e ci da una buona indicazione per stabilire dei bound sul valore della funzione obiettivo del problema di PL intera: questo è importante perché nonostante non ci fornisca una soluzione che è implementabile e il cui arrotondamento potrebbe non essere ammissibile o non ottimale, ci può indicazioni per stabilire la qualità delle soluzioni intere che via via troveremo con l'algoritmo del B&B. Se ho un problema di massimizzazione il valore della funzione obiettivo nel punto di ottimo del problema rilassato costituirà un upper bound per il problema di PL intera; se sto minimizzando il valore ottimo costituirà un lower bound per il problema di PL intera.

Funzionamento dell'algoritmo del Branch-And-Bound

$$\begin{aligned}
 & \text{MAX : } 2X_1 + 3X_2 \\
 & \text{S.t. : } X_1 + 3X_2 \leq 8.25 \\
 & \quad 2.5X_1 + X_2 \leq 8.75 \\
 & \quad X_1 + X_2 \geq 0 \text{ and integer}
 \end{aligned} \tag{10.1}$$

La regione ammissibile del problema rilassato sarà (quella in blu):

Solution to LP Relaxation

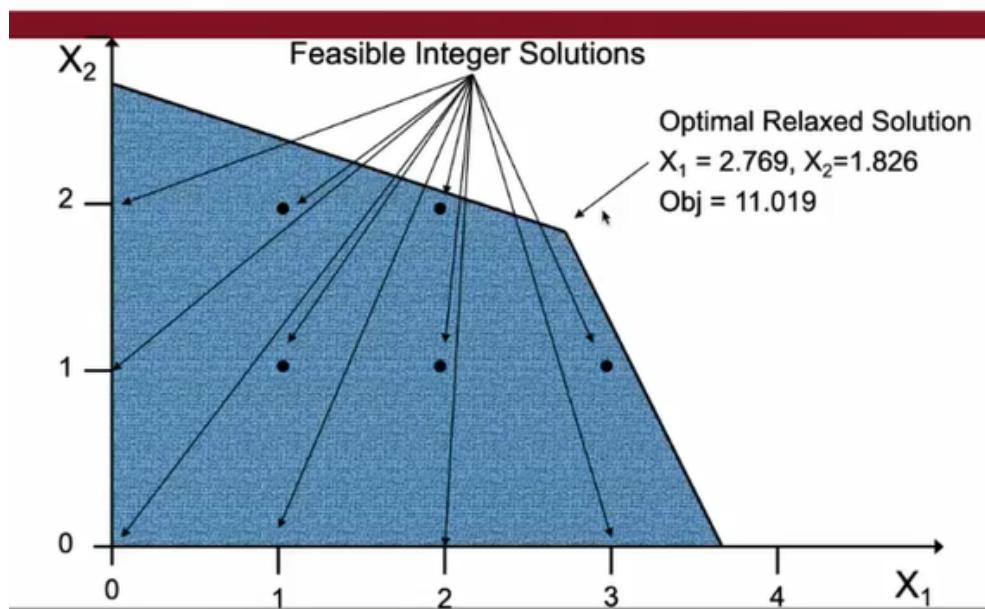


Figure 10.8: LP relaxation

I puntini saranno le soluzioni intere ammissibili. Il metodo del B&B parte dalla soluzione ottima del problema rilassato: in questo caso risolve il problema con il metodo del simplex e ottiene la soluzione

$X_1 = 2.769$ e $X_2 = 1.826$ con una funzione obiettivo pari a 11019. A questo punto noi vediamo che la soluzione ottima è nel vertice indicato dalla freccia: si tratta di una soluzione **non intera** e ovviamente il nostro problema è che l'algoritmo del simplex trova la soluzione ottima sui vertici. Quello che noi vogliamo fare sarebbe partizionare la regione in modo tale da ottenere le regioni ammissibili dove il vertice sia un punto intero.

Il primo passo è quello del **branching**: in questa fase, l'algoritmo genera due sottoproblemi eliminando una parte della regione ammissibile che non contiene soluzioni intere, scegliendo una variabile su cui fare il branching. In questo caso sia X_1 che X_2 non sono intere. La variabile su cui fare branching potrà essere scelta allora fra X_1 o X_2 . Cosa significa **fare branching**? Il concetto alla base è quello del **DIVIDE ET IMPERA**: divide il problema in sottoproblemi in modo tale che continuando a generare questi sotto problemi prima o poi genero un sottoproblema dove ho la soluzione ottima che si trova su un vertice che ha un valore intero. Come suddivido questo problema? Con l'operazione di branching che può essere applicato a qualsiasi variabile che abbia un valore ottimo del problema rilassato e non intero.

Supponiamo di considerare la variabile X_1 per fare il branching. X_1 ha un valore compreso fra 2 e 3. Fare branching significa generare due sottoproblemi imponendo che in un caso la variabile X_1 sia minore/uguale a 2 e nell'altro caso la variabile X_1 sia maggiore/uguale a 3. Vado quindi ad eliminare tutti i punti compresi fra 2 e 3 esclusi (non li elimino), ovvero tutti quelli non interi compresi fra questi due estremi. Sicuramente eliminando questi punti non elimino soluzioni intere. Eliminandoli dalla regione ammissibile non comprometto il fatto di trovare nel resto della regione la soluzione ottima intera.

Supponendo quindi di aver scelto X_1 come variabile di branching, genero due sottoproblemi:

- In uno aggiungerò il vincolo $X_1 \leq 2$;
- Nell'altro aggiungerò il vincolo $X_1 \geq 3$;

The Branch-And-Bound Algorithm

Problem I MAX: $2X_1 + 3X_2$
 S.T. $X_1 + 3X_2 \leq 8.25$
 $2.5X_1 + X_2 \leq 8.75$
 $X_1 \leq 2$
 $X_1, X_2 \geq 0$ and integer

Problem II MAX: $2X_1 + 3X_2$
 S.T. $X_1 + 3X_2 \leq 8.25$
 $2.5X_1 + X_2 \leq 8.75$
 $X_1 \geq 3$
 $X_1, X_2 \geq 0$ and integer

Figure 10.9: LP relaxation

Quindi abbiamo preso la regione ammissibile che è stata divisa in due sottoproblemi e avremo come risultato grafico il seguente:

Solution to LP Relaxation

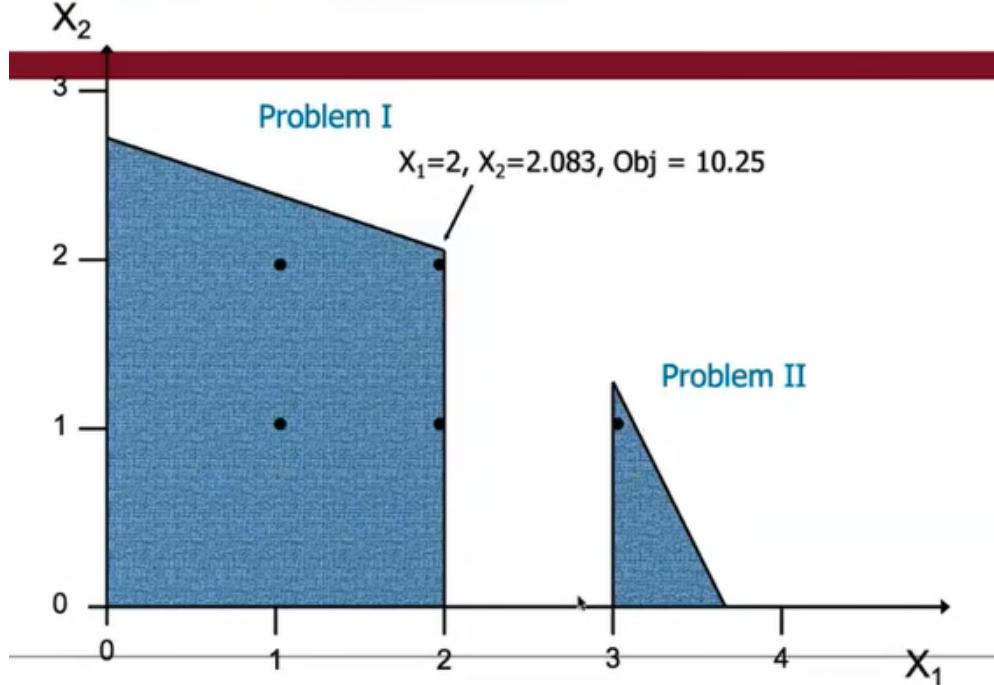


Figure 10.10: Branching 1

Nonostante abbia eliminato dei punti dalla regione ammissibile del problema rilassato, non è stato eliminato nessun punto intero all'interno di questa regione ammissibile. Quindi la soluzione del problema di PL intera si trova o da una parte o dall'altra, devo cercarla quindi in questi due sottoproblemi.

Ovviamente devo scegliere un problema dal quale fare branching: in questo caso abbiamo iniziato dal problema 1, lo abbiamo risolto ottenendo $X_1 = 2$ e $X_2 = 2.083$ con funzione obiettivo 10.25. Quindi procediamo con la variabile X_2 , generando altri due sottoproblemi: aggiungendo da una parte il vincolo $X_2 \leq 2$ e dall'altra il vincolo $X_2 \geq 3$, la quale aveva un valore compreso fra 2 e 3:

The Branch-And-Bound Algorithm

Problem III

$$\begin{aligned} \text{MAX: } & 2X_1 + 3X_2 \\ \text{S.T. } & X_1 + 3X_2 \leq 8.25 \\ & 2.5X_1 + X_2 \leq 8.75 \\ & X_1 \leq 2 \\ & X_2 \leq 2 \\ & X_1, X_2 \geq 0 \text{ and integer} \end{aligned}$$

Problem IV

$$\begin{aligned} \text{MAX: } & 2X_1 + 3X_2 \\ \text{S.T. } & X_1 + 3X_2 \leq 8.25 \\ & 2.5X_1 + X_2 \leq 8.75 \\ & X_1 \leq 2 \\ & X_2 \geq 3 \\ & X_1, X_2 \geq 0 \text{ and integer} \end{aligned}$$

Figure 10.11: Branching 2

Il sottoproblema III con $X_2 \leq 2$ ha una regione ammissibile pari a quella a sinistra:

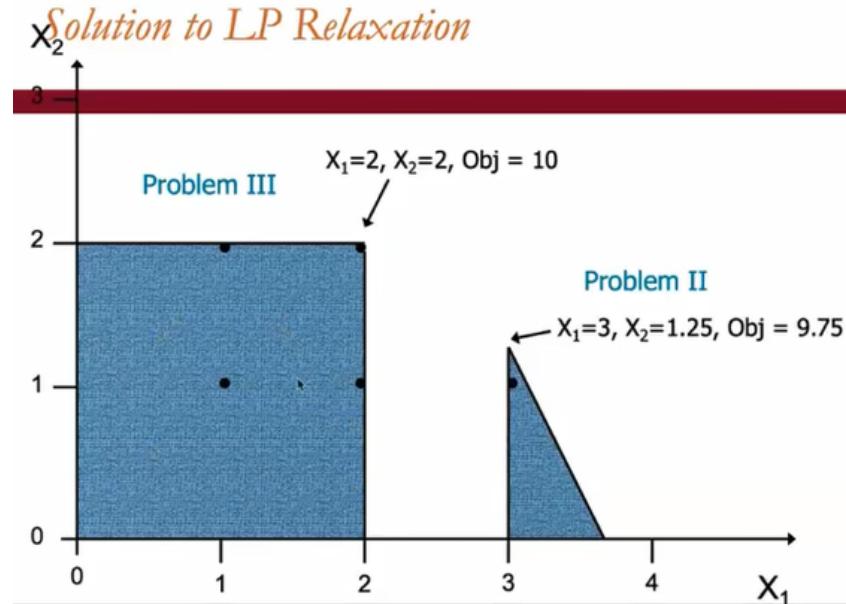


Figure 10.12: Branching 2

Mentre il sottoproblema IV con $X_1 \leq 2$ e $X_2 \geq 3$ **non è ammissibile**, infatti mettendo un vincolo dove $X_2 \geq 3$ vediamo che l'intersezione tra la regione ammissibile e questo ulteriore vincolo è una regione vuota e il problema non conterrà soluzioni ammissibili.

A questo punto ho il problema III e il problema II che devono essere ancora analizzati. Prendiamo il sottoproblema II e prendiamo la soluzione ottima che sarà $X_1 = 3$ e $X_2 = 1.25$. Essendo X_2 non intera possiamo fare branching su X_2 . Andremo a generare i sottoproblemi V e VI, aggiungendo il vincolo $X_1 \leq 1$ e $X_2 \geq 2$. Sostanzialmente quello che succede è che continuiamo a fare branching sui nostri sottoproblemi in modo tale da arrivare ad avere una soluzione ottima in una sottoregione che coincide con un punto intero.

B&B Summary

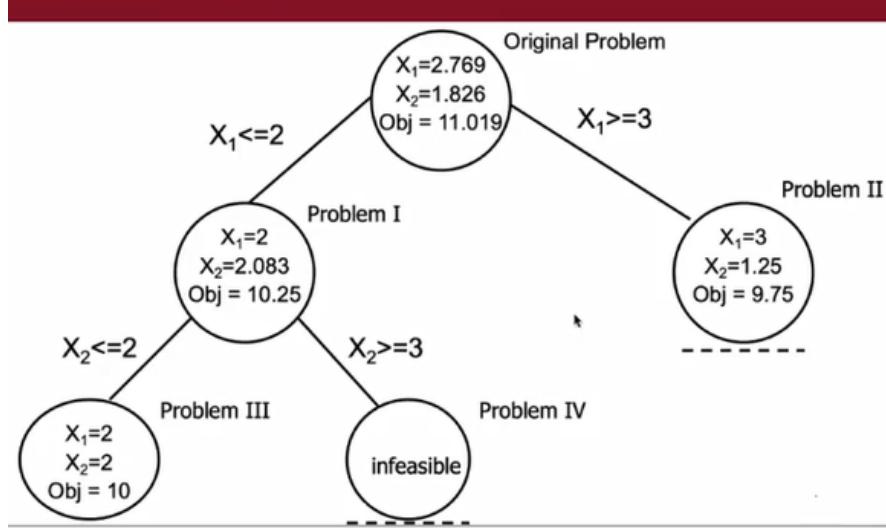


Figure 10.13: Branching final

Però anche questo approccio ci porterebbe ad un numero di sottoproblemi molto elevato con più variabili da considerare.

Alla fase di branching si unisce la fase di **bounding**: se ho risolto il problema III con X_1 e X_2 interi, ho trovato una soluzione ammissibile, perché intera e vale 10. Quello che abbiamo detto all'inizio era:

con il metodo del branch-and-bound, una volta che ho trovato una soluzione ammissibile posso impiegarci molto a dimostrare che è tale. Questa soluzione intera che ho trovato posso usarla per valutare la **qualità** delle soluzioni che potrei trovare negli altri sottoproblemi. Quindi, per esempio, in questo caso ho una soluzione intera che ha una funzione obiettivo di 10. Se risolvo il sottoproblema II vedo che il valore ottimo della funzione obiettivo è 9.75 e non è ammissibile, nel senso che la soluzione non è intera. So però che il valore 9.75 costituirà un upper bound delle soluzioni intere contenute in questa regione: ciò significa che qualsiasi soluzione intera contenuta in questa regione non potrà mai superare il valore di 9.75. Quindi, la domanda che mi pongo è: **mi conviene fare branching su X2 e risolvere il sottoproblema V e il sottoproblema VI andando ad esplorare questa regione?** Assolutamente no! Posso dire che questa sottoregione non la esploro perché comunque otterrei delle soluzioni che sarebbero peggiori rispetto alla soluzione che ho già trovato. Quindi il problema non viene in realtà investigato e non faccio branching su X2 e non genero i problemi V e VI.

A questo punto posso analizzare la situazione e per farlo sintetizzo il percorso fatto con l'algoritmo del branch-and-bound con un albero, l'**albero del branch-and-bound**. Proseguendo nella risoluzione dei sottoproblemi l'**upper bound** diminuisce, come vediamo nel grafico ad albero del branch-and-bound. Trovata la soluzione intera l'algoritmo si blocca: saprò che la mia soluzione ottima sarà compresa fra 10 che è una soluzione intera ammissibile che ho trovato (non so ancora se è una soluzione ottima, ma potrebbe esserlo e sarebbe da dimostrare e costituisce un lower bound) e 10.25. Andando a guardare il valore della funzione obiettivo del problema 2 vedo che è pari a 9.75, valore peggiore della soluzione di 10 della soluzione ammissibile trovata. E quindi posso dire che la porzione di regione ammissibile riferita al sottoproblema II non vale la pena esplorarla perché non vi troverò valori maggiori di 9.75. Terminerò quindi il nodo.

Vediamo che con l'algoritmo del branch-and-bound continuerò a generare sottoproblemi finché:

- un sottoproblema non sarà più ammissibile e quindi mi fermo;
- il sottoproblema ha una soluzione intera e quindi mi fermo perché ho trovato una soluzione intera;
- oppure perché il valore dell'upper bound del sottoproblema è peggiore del valore della funzione obiettivo di una soluzione intera migliore che ho già trovato in passato risolvendo i precedenti sottoproblemi.

Quali domande ci vengono in mente dopo aver visto questo schema risolutivo ad albero? Ho fatto delle scelte, partendo da X1, ma avrei potuto scegliere di fare branching partendo da X2; oppure quando ho i sottoproblemi aperti ho scelto di processare prima il problema I, ma avrei potuto farlo prima col problema II generando due sottoproblemi, anche se poi avrei comunque dovuto esplorare il problema I.

Quindi quando metto in pratica il branch-and-bound devo mettere in pratica una serie di strategie che devo determinare. Ovviamente quello che succede è che nella variabile di branching la possibile strategie di scelta sono:

- In ordine di indice: parto da X1 perché ha l'indice minore;
- Nel caso in cui io abbia delle variabili binarie, posso partire dalla variabile che ha un valore reale più distante dai valori interi (ogni tanto può funzionare come strategia)

Qualsiasi scelta è una euristica: non esiste una scelta strategica della sequenza con cui faccio branching che sicuramente funziona meglio di altre. Ogni problema può funzionare meglio con l'una o con l'altra strategia. Se un problema ci impiega molto con una, posso provare a risolverlo con l'altra.

L'altra scelta strategica è quella di come esplorare l'albero: qual è il sottoproblema che scelgo di esplorare per primo? In genere la scelta che si fa è quella di scegliere il problema **più promettente**. Fra i due scelgo quella con la funzione obiettivo più alta, avendo la speranza di trovare delle soluzioni intere più elevate che mi permettano di evitare l'esplorazione di altre regioni più inutili risparmiando tempo

computazionale.

Vediamo di riassumere l'**algoritmo del Branch-and-Bound**:

- **Inizializzazione:** si rilassano le condizioni di interezza del problema di PL intera e si risolve il problema di PL con un algoritmo come il simplex;
- Se il problema è un problema di massimizzazione devo settare il **lower-bound** definito **Z-best** che rappresenta la soluzione intera ammissibile migliore trovata fino a questo momento che all'inizio sarà pari a $Z_{best} = -\infty$. Se il problema è di minimizzazione si avrà $Z_{best} = \infty$. Quello che si fa è avere un limite che mi dice l'upper bound della mia soluzione ottimale (che mi dice che più di quel valore non si può ottenere); questo limite superiore mi dice che qualunque valore inferiore non lo considererò a prescindere. L'importanza dell'upper bound è quello di capire la qualità delle soluzioni intere che trovo. L'importanza del lower bound è che risparmio tempo computazionale perché non vado ad esplorare le regioni che non mi darebbero un risultato migliore.

In generale lo Z_{best} rappresenta il valore della funzione obiettivo della soluzione intera migliore conosciuta aggiornata all'ultimo passo dell'algoritmo.

- **Branching:** genero diversi sottoproblemi dato un problema da cui partire. In un caso aggiungo il vincolo $X_j \leq INT(b_j)$ (ovvero dell'intero inferiore) e dall'altro sottoproblema $X_j \geq INT(b_j)$ (dove l'intero è quello superiore). Metto questi problemi candidati ad essere risolti in una lista di problemi candidati ad essere risolti;
- Se la lista dei possibili candidati è vuota allora ci si ferma perché si sono esplorati tutti i sottoproblemi, altrimenti devo scegliere un problema dalla lista e qui entra in gioco la strategie: posso scegliere il problema dalla lista con una strategia *first in-first out* oppure *last in-first out*, oppure scelgo rispetto al valore della funzione obiettivo che il sottoproblema aperto ha. Queste sono strategie che si implementano con il software. Quando si risolvono a mano si parte dal problema con la funzione obiettivo migliore, perché mi permette di trovare con maggiore probabilità la soluzione che mi fornisce la soluzione ottimale. Quindi prendo uno dei sottoproblemi, rilasso le condizioni di interezza e lo risolvo.
 - Se non esiste una soluzione del sottoproblema, allora non è ammissibile e passo al prossimo e ne prendo un altro dalla lista dei candidati; Z_{cp} denota allora il valore ottimo della funzione obiettivo del problema candidati corrente (dove cp sta per *current problem*): sarà il valore della funzione obiettivo del sottoproblema che sto risolvendo;
 - se Z_{cp} non è migliore (è peggiore) di Z_{best} (che corrispondeva ad una soluzione intera che ho già trovato - cioè, non migliore significa che se ho un problema di massimizzazione vuol dire che Z_{cp} è minore/uguale di Z_{best} cioè ha un valore peggiore della soluzione intera che ho già trovato; se ho un problema di minimizzazione Z_{cp} sarà maggiore/uguale a Z_{best}), allora non procedo, lo elimino e passo ad un altro problema candidati;
 - se invece la soluzione del sottoproblema ha una funzione obiettivo migliore di Z_{best} e la soluzione che ottengo non è intera, **devo fare branching** scegliendo una variabile non intera e generare due sottoproblemi da inserire nella liste dei sottoproblemi candidati ad essere risolti;
 - se la soluzione di questo problema è migliore di Z_{best} ed è intera, allora Z_{best} diventa uguale a Z_{cp} .
 - una volta chiusi tutti i sottoproblemi ci si ferma perché significa che si è trovati la soluzione migliore che corrisponde a Z_{best} che è la soluzione migliore intera trovata.

Chapter 11

LEZIONE 9 - Practical Session - Integer Programming (14/04/2021)

La programmazione lineare intera è **NP-completo**: la **NP-completezza** (NP = non-polynomial) è una classe di difficoltà algoritmica che implica che il problema è estremamente difficile da risolvere (il problema lineare con variabili continue è P-completo, ovvero può essere risolto in tempo polinomiale, ovvero significa che rispetto al numero delle variabili esiste una funzione che è in grado di risolvermi il problema in un tempo che è proporzionale al numero delle variabili elevate per un certo esponente). Per la programmazione intera/mista questo non accade. Essi sono problemi estremamente più difficili da risolvere: richiede una quantità di tempo che spesso è esponenziale rispetto al numero di variabili decisionali coinvolte nel modello.

Partiamo da un esempio che dovrebbe farci capire quali sono gli elementi principali quando parliamo di programmazione intera e che differenza c'è tra una soluzione intera e una continua pur nella medesima descrizione del modello (immaginiamo di dover risolvere il problema con le variabili x e y intere e una volta continue):

$$\begin{aligned} & \text{MAX : } y \\ & \text{S.t. : } -x + y \leq 1 \\ & \quad 3x + 2y \leq 12 \\ & \quad 2x + 3y \leq 12 \\ & \quad x, y \geq 0 \\ & \quad x, y \in \mathbb{Z} \end{aligned} \tag{11.1}$$

Guardiamo subito al dominio del nostro problema per quanto riguarda il caso continuo e quello intero: nel caso intero tutte e sole le soluzioni del nostro problema, ovvero quella coppia di x , y che verificano questi vincoli sono rappresentate dai punti rossi che sono allineati rispetto alle ascisse/ordinate intere. Soltanto uno dei punti rossi può essere il nostro punto ottimale:

The following figure depicts both the feasible domain and the LP relaxation. The feasible integer points are shown in red, and the red dashed lines indicate their **convex hull**, which is the **smallest polyhedron** that contains all of these points.

The blue lines together with the coordinate axes define the polyhedron of the **LP relaxation**, which is given by **the inequalities without the integrality constraint**.

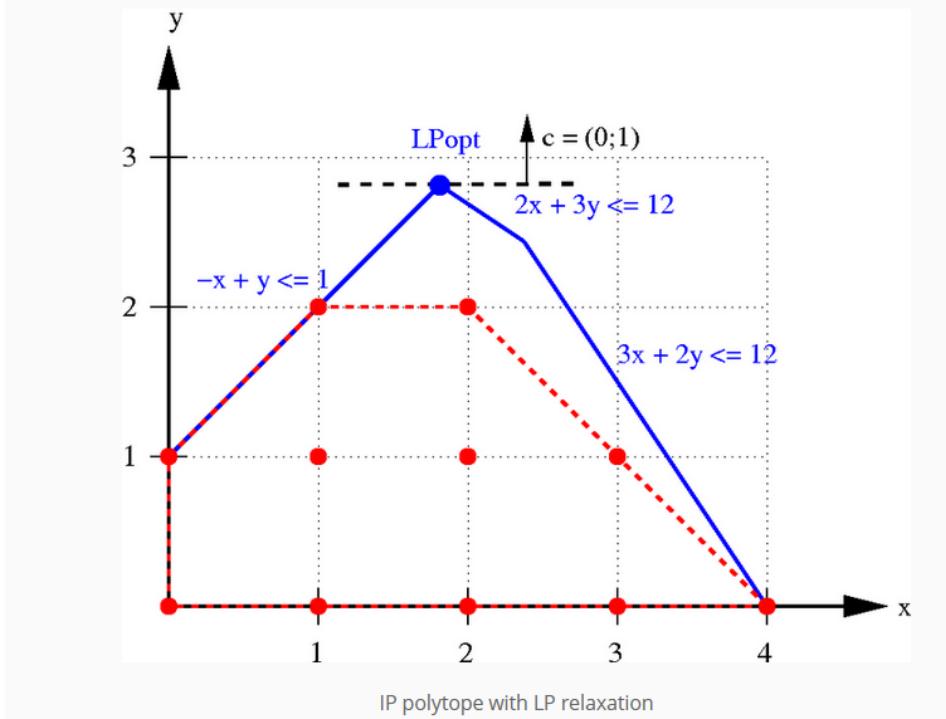


Figure 11.1: Integer programming

Se il problema viene rilassato, otteniamo un nuovo dominio che è definito da queste linee blu: avremo valori positivi di y : tutte le soluzioni ammissibili del nostro problema sono continuamente distribuite all'interno di questa area. Per quanto riguarda invece l'area continua definita però dai punti interi (quella al di sotto delle linee tratteggiate rosse) viene chiamata **convex hull** e definita come: **il più piccolo poliedro o politopo che contiene i punti del nostro dominio**. E' importante tale definizione in quanto se noi conoscessimo il **convex hull**, se avessimo una maniera per poterlo definire in maniera precisa, potremmo utilizzare la definizione del convex hull, ovvero i vincoli del convex hull, al posto dei vincoli del problema originale, quindi rilassare il problema ed essere sicuri che la soluzione del problema rilassato sul convex hull concide con la soluzione ottima del problema originale. Infatti, siccome stiamo massimizzando la variabile y , il valore ottimo per la nostra funzione obiettivo nel caso del convex hull o comunque dello spazio delle soluzioni intere è costituita dai punti rossi più alti che rappresentano i vertici del convex hull e per questo sarebbero ottimi.

Il problema è che non siamo spesso in grado di farlo. Quello che siamo in grado di fare è utilizzare un rilassamento continuo delle nostre variabili risolvendo il problema in blu. In questo modo, otteniamo la soluzione che vediamo in blu più in alto di tutti. Quel vertice però non è un vertice con coordinate intere, ma le sue coordinate sono frazionarie. Quello che possiamo fare è dire che quel vertice rappresenta per il problema originario un **upper bound** in quanto l'area del nostro spazio delle soluzioni feasible è più grande e contiene quella delle soluzioni intere.

Dopo aver osservato questa cosa (ovvero che la soluzione ottima del problema rilassato è un upper bound nel caso di massimizzazione) osserviamo che l'arrotondamento di questa soluzione all'intero più vicino (in questo caso sarebbe il punto $(2,3)$) non sempre coincide con una soluzione ammissibile per il problema originale).

Vediamo una piccola introduzione dell'algoritmo del Branch-and-Bound: tale algoritmo permette una

enumerazione sistematica delle possibili soluzioni candidate per un certo problema dividendo di volta in volta lo spazio delle soluzioni ammissibili in sottospazi. Questi sottospazi devono essere il più possibile enumerati in **maniera implicita**: si deve subito dire se questo sottospazio contiene una soluzione ottima, oppure siamo sicuri che la soluzione ottima non è contenuta in tale spazio. Se possiamo dire questo, allora saremo in grado di affermare di non voler analizzare un certo numero di sottospazi perché sicuramente non contengono la soluzione ottima. Saremo in grado di scartare un numero potenzialmente infinito di soluzioni che non dovremo risolvere.

Vediamo quali sono i passi di un Branch-and-Bound generico (generico = perché l'implementazione dell'algoritmo è soggetta ad alcune scelte architettoniche: ad esempio, a come suddividere lo spazio delle soluzioni oppure priorizzare la scelta dei sottospazi da analizzare rispetto a tutti gli altri. In base a queste scelte ottengo un implementazione di questo algoritmo diversa da un'altra e che abbia maggiore successo per la risoluzione di un problema per una certa formalizzazione piuttosto che per un'altra):

- Partiamo dal nostro modello $P_1 = P$; Settiamo il valore della variabile $Z* = -\infty$ (stiamo supponendo che vogliamo massimizzare) e $Z*$ vogliamo che rappresenti il miglior valore della funzione obiettivo per una soluzione ammissibile (che verifichi i vincoli del nostro problema) e che pertanto sia intera. Tale soluzione, se esiste, ci permetterà di settare $Z*$ ad un valore diverso da $-\infty$. Una tale soluzione **migliore intera** trovata fin'ora viene chiamata **soluzione incombente**. Se non la conosciamo fin dall'inizio possiamo partire dall'idea di impostare $Z*$ uguale a $-\infty$.
- A questo punto prendiamo il nostro problema chiamato **problema radice** o **nodo radice** e generiamo una stima ottimistica di $Z*$ rilassando il nostro problema originale intero: consideriamo le sue variabili continue e a quel punto abbiamo un nuovo valore della funzione obiettivo. Tale valore $Z*$ è una stima ottimistica (cioè è un upper bound per la possibile soluzione ottima per il nostro problema). A quel punto possiamo chiederci se la soluzione del problema rilassato è intera: se lo fosse, possiamo fermarci perché è anche ottima; se non lo fosse, allora il valore della stima ottimistica di $Z*$ (quindi l'upper bound) lo chiamiamo Z_1 che ci darà un'indicazione su tutti i possibili sottospazi: cioè non potremmo avere nessuna soluzione intera per il nostro problema con valore della funzione obiettivo che sia maggiore di Z_1 . Per cui esso limita il valore che la funzione obiettivo potrà assumere in tutti i possibili sottospazi che si possono generare a partire dal nodo radice;
- A questo punto siccome il problema non l'abbiamo ancora chiuso (stiamo supponendo che la soluzione trovata del problema rilassato non sia intera), eseguiamo uno **step di branching** che ci dice di prendere uno spazio che ancora non abbiamo scartato (in questo caso abbiamo solo il nodo radice): il nodo radice viene splitto in T sottospazi. Come? Ci possono essere vari modi, ma l'unica accortezza è che questi sottospazi siano tra loro disgiunti e che la loro unione coincida con lo spazio di partenza (quindi con P_i), ovvero:

$$P_{ij} \quad j = 1, \dots, t \mid U_j P_{ij} = P_i \quad (11.2)$$

Come è possibile eseguire tale split?

Risolviamo il nostro problema P_i in maniera rilassata; avremo un certo numero di variabili non intere; prendiamo una di queste variabili, ad esempio x_p che assume valore $s_i[p]$, quindi un valore non intero relativo alla variabile p – esima e in questo caso possiamo creare due sottoproblemi imponendo i seguenti vincoli:

- In uno si impone il vincolo che $x_p \leq s_i[p]$ (cioè dell'intero inferiore di $s_i[P]$);
- Nell'altro caso che $x_p \geq s_i[p] + 1$ (cioè il più vicino intero superiore).

Abbiamo quindi aggiunto un vincolo in un problema e un vincolo in un altro. E abbiamo creato due problemi il cui dominio è spezzato, ma che insieme costituiscono il dominio del problema originale. Questo è un esempio possibile di branching.

- Bounding step: per ognuno di questi sottoproblemi risolviamo il loro rilassamento continuo e calcoliamo il valore della funzione obiettivo del problema rilassato. Questo valore rappresenta per ognuno dei sottoproblemi il valore migliore che la funzione obiettivo potrà ottenere per il problema intero in quel sottospazio. Ci domandiamo: è questo valore peggiore di un valore intero che io ho già trovato? Se è così è inutile analizzare ulteriormente quel sottospazio, perché so già che al massimo otterrò il valore Z . Questa fase viene chiamata **Fathoming**;
- Fathoming step: ovvero, eliminazione o il pruning di un certo ramo del branch-and-bound. Ciò significa che quando mi occupo di questa fase mi devo chiedere: il mio sottoproblema ha soluzione? Se è infeasible (cioè non ha soluzione) non devo più considerarla. Se ha soluzione ed è intera, vediamo il valore della sua funzione obiettivo: se è migliore del valore della funzione obiettivo della soluzione incombente (quindi se è migliore della soluzione trovata fin'ora) allora cambierò il valore della soluzione incombente e quindi il valore di Z^* . Non dovrò più considerare ulteriore suddivisione del sottospazio per quel nodo perché la mia soluzione è già intera.

Infine, se ci troviamo nel caso in cui la variabile Z_{ij} , quindi il valore della funzione obiettivo è minore del valore intero migliore trovato fin'ora, allora sicuramente non dovrò più analizzare quel sottospazio perché in quel sottospazio non potrà esistere una soluzione migliore della soluzione incombente.

- Infine eseguiamo il **test dell'ottimalità**: se non ci sono più nodi aperti nella nostra lista dei nodi aperti e abbiamo una soluzione incombente ovvero intera, allora quella soluzione trovata durante il processo è sicuramente ottima; se non ci sono più nodi e non abbiamo trovato nessuna soluzione intera per il nostro problema allora il nostro problema è infeasible, ovvero non ha soluzioni ammissibili. Altrimenti se esistono ancora dei nodi che non abbiamo ancora esplorato, dobbiamo tornare al **punto 3** e ricominciare con il branching dividendo lo spazio in sottospazi e ripetere l'analisi.

Per capirne meglio il funzionamento guardiamo al seguente esempio:

Example

Let's consider the following example taken from [link](#).

$$\begin{aligned} \text{max } & 5x_1 + 8x_2 \\ \text{subject to } & x_1 + x_2 \leq 6 \\ & 5x_1 + 9x_2 \leq 45 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

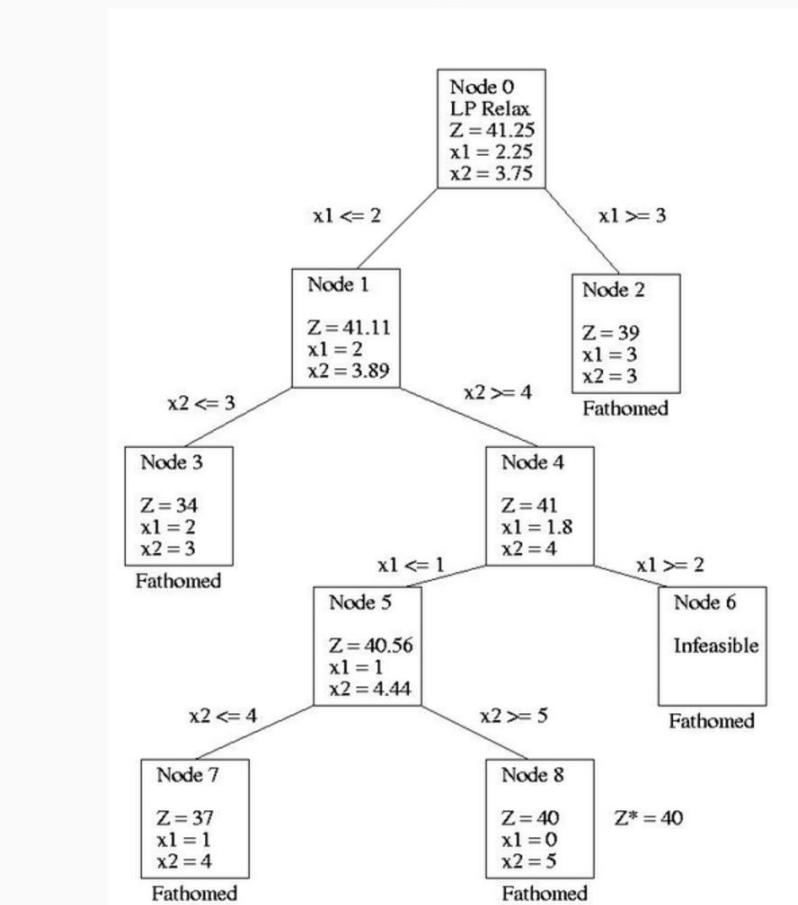


Figure 11.2: Branch-and-Bound example

In questo caso ci troviamo a dover massimizzare la funzione:

$$5x_1 + 8x_2 \quad (11.3)$$

Quindi, prima di tutto dobbiamo risolvere il problema **rilassato**, cioè imponendo che le variabili x_1, x_2 non siano più intere ma continue, ovvero in \mathbb{R} . Facendo questo noi possiamo usare l'algoritmo del simplex: ci permette di ottenere una soluzione in breve tempo. Quindi:

- Rilasso il problema;
- Calcolo il valore della funzione obiettivo \mathbb{Z} nei vari passaggi;

La prima soluzione ottenuta non è intera, quindi il valore di Z pari a 41.25 rappresenta un **upper bound** per l'intero problema: la mia soluzione ammissibile o ottima intera non potrà avere valore della funzione obiettivo maggiore di 41.

A questo punto applichiamo il **branching** scegliendo una delle due variabili, come x_1 creando due sottoproblemi: in uno creiamo il vincolo che $x_1 \leq 2$; nell'altro $x_1 \geq 3$, spezzando il problema originale, la cui unione dei due domini corrisponde al dominio iniziale.

Dobbiamo scegliere una politica di analisi di questi nodi: possiamo decidere sempre di analizzare i nodi sullo stesso livello insieme e di scegliere sempre il nodo più a sinistra. Scegliamo quindi prima il nodo 1: lo rilassiamo e lo risolviamo. Otteniamo che Z è minore rispetto a prima e che x_1 è intero ma non x_2 . Tale nodo non può essere quindi chiuso. Inoltre, il valore della sua funzione obiettivo non è minore di Z^* che è pari a -inf perché non siamo ancora stati in grado di trovare nessuna soluzione intera. A questo punto questo nodo rimane in sospeso. Dovremo risolvere il nodo allo stesso livello ma a più a destra. Si risolve il nodo II ottenendo una soluzione intera. Il valore della funzione obiettivo è 39 che rappresenta la nostra **soluzione incombente** perché è la prima soluzione intera che abbiamo trovato. Questo vuol dire che d'ora in poi siamo interessati a cercare soluzioni con valore della funzione obiettivo **maggiore di 39**, ma ovviamente minore di 41 che rappresenta il nostro bound superiore. Qualsiasi cosa minore di 39 non interessa.

Creiamo poi due nuovi problemi dal nodo I: imponiamo che $x_2 \leq 3$. In tal modo creiamo un nuovo problema che sì è una soluzione intera, ma il valore della funzione obiettivo è minore di 39, ovvero dell'attuale **soluzione incombente**. Quindi la scartiamo e non continuiamo ad analizzare il sottospazio relativo al nodo 3.

Analizziamo il nodo IV e otteniamo Z pari a 41 il valore di x_2 intero e il valore di x_1 no. Essendo il valore di Z maggiore di 39 ovvero dell'attuale soluzione incombente possiamo analizzare ulteriormente tale sottospazio. Prendiamo la variabile x_1 che è frazionaria e creiamo due nuovi vincoli. Il nodo VI sarà *infeasible* (con $x_1 = 2$ non è possibile, sostituendo, soddisfare tutti i vincoli contemporaneamente), ovvero il dominio che si ottiene a partire dal problema iniziale e aggiungendo i vincoli che portano al nodo VI (uno spazio in questo caso costituito da un solo punto), non è possibile trovare una soluzione al nostro problema e non è quindi ammissibile, ovvero è infeasible.

L'unico nodo aperto è il nodo 5 e continuiamo ad analizzarlo. L'ultimo nodo disponibile è il nodo VIII con x_1 e x_2 interi e valore di $Z = 40$: tale soluzione è migliore della soluzione presente al nodo II, avendo il valore della funzione obiettivo maggiore della precedente soluzione incombente.

RICORDA: ogni volta che faccio branching la mia volontà è quella di ridurre il più possibile l'area del problema rilassato verso quella della **convex hull**, cercando di approssimare in maniera continua lo spazio discreto, aggiungendoci sempre più vincoli.

The Case Study Problem

Questo è il problema successivo a quello precedente appartenente a *Ex. 8 - The Facility Location Problem*.

Tale esercizio insiste più sulla problematica di modellazione del problema che non sul B&B. Per noi ora il B&B è uno strumento come il simplesso che verrà implementato dal software/solver. Per ora è importante **modellare correttamente il problema**.

Tale *case study* era un assignment dato due anni fa ed è relativo al problema cosiddetto di *facility location*: si tratta del problema di posizionare un gruppo di facility (edifici come fabbriche o un qualcosa che dà un servizio). Tali facility devono distribuire un certo bene ad un gruppo di clienti che ne hanno bisogno: ad esempio, distribuzione di vaccini. La domanda è: *dove posizionare questi hub per minimizzare i costi di distribuzione e anche di costruzione degli hub che devo costruire? Allo stesso tempo verificare una serie di altri vincoli, per esempio relativi al budget della costruzione e al numero massimo di hub da*

costruire?

Supponiamo una serie di punti:

- Le possibili *locations* per gli hub sono **conosciuti**: abbiamo un certo numero intero di posizioni su una mappa su cui possiamo costruire la nostra facility;
- Conosciamo i costi di costruzione dei nostri hub che possono essere **fissi** ed essere uguali per tutte le facility oppure possono dipendere dalla particolare facility perché costruire in una zona di pianura non ha gli stessi costi che costruire la struttura in una zona di montagna;
- Vi sono i **costi di trasporto** che supponiamo noti e *proporzionali* alla distanza tra la facility e l'utente finale: più è lontano il cliente e maggiore il costo di trasporto;
- Eventualmente un solo hub è in grado di fornire tutti i clienti (ha sufficienti prodotti per rifornire tutti i *customers*);

Si deve quindi decidere:

- Quante facility costruire?
- Dove costruirle?
- Assegnare ad ogni facility un certo numero di clienti;

Il problema può essere formulato come un problema di **programmazione intera/mista**. La successiva risoluzione avviene alla relativa pagina del problema, alla sezione *Data*.

Riguardo la **formulazione** del problema:

- **Variabili decisionali**: la prima variabile sarà data da y_j che rappresenta l'insieme delle possibili facilities che ci dice se costruisco o non costruisco la facility in quel punto ($\forall j \in 1, \dots, m$); L'altra variabile è $x_{i,j}$ che ci dice se un certo cliente è associato ad una certa facility, dove i è il cliente e j è la facility ($\forall i \in 1, \dots, n$ and $\forall j \in 1, \dots, m$), creando quindi una relazione fra il customer i e la facility j . Anche questa è una variabile binaria: o esiste o non esiste questa relazione.

C'è però un vincolo a cui pensare, ovvero il seguente:

$$x_{i,j} \leq y_j \quad (11.4)$$

Ovvero: l'idea è che posso associare un cliente alla facility **solo se** quella facility esiste, cioè se è stata costruita. Se $x_{i,j}$ è 1 allora per forza y_j ; al contrario se y_j è uguale a 0 allora tutte le $x_{i,j}$ devono essere uguali a 0. Se si decide che deve esistere un vincolo tra una facility e un cliente allora quella facility deve essere stata costruita; viceversa, se quella facility NON è stata costruita non può sicuramente esistere nessun vincolo con i clienti circostanti.

- **Funzione obiettivo**: la vogliamo **minimizzare**. Vogliamo minimizzare **due costi**:

- il costo relativo al trasporto: dipende dalla distanza che possiamo considerarla come una costante, ovvero un coefficiente già calcolato;
- il costo relativo alla costruzione: essi sono un dato del nostro problema.

Quello che vogliamo fare quindi per tutte le facility e per tutti i customer moltiplichiamo il costo relativo alla distanza al valore della variabile $x_{i,j}$: quindi se c'è quella relazione fra il cliente i e la facility j allora devo tenere conto di quella componente della distanza. Riguardo ai costi è più facile: se y_j è uguale a 1 cioè se ho deciso di costruire la facility in quel punto, devo tenere conto dei costi di costruzione, altrimenti li devo moltiplicare per 0 e questo termine non avrà effetto sul costo totale della nostra soluzione.

$$\sum_{i=1}^n \sum_{j=1}^m distance(i, j) \times x_{i,j} + \sum_{j=1}^m building\ costs(j) \times y_j \quad (11.5)$$

- Un customer deve essere servito **solo da una facility**: se il mio customer viene servito dalla facility 1, allora non potrà essere servito dalla facility k . Questo significa che se sommo rispetto a tutte le possibili facility per un certo cliente, la somma delle $x_{i,j}$ deve essere uguale a 1. (perché sarà servito solo da una facility). Ovvero:

$$\sum_{j=1}^m x_{i,j} = 1 \quad \forall i \quad (11.6)$$

Non minore perché vorrebbe significare che è valido il caso in cui il cliente non è servito da nessuno; né maggiore di 1 perché vorrebbe dire che il cliente è servito da più di una facility.

Mettendo tutto insieme otteniamo il modello seguente:

$$\begin{aligned} MIN : & \sum_{i=1}^n \sum_{j=1}^m distance(i, j) \times x_{i,j} + \sum_{j=1}^m building\ costs(j) \times y_j \\ S.t. : & \sum_{j=1}^m x_{i,j} = 1 \quad i = 1, \dots, n \\ & x_{i,j} \leq y_j \quad i = 1, \dots, n \quad j = 1, \dots, m \\ & x_{i,j} \in (0, 1) \quad i = 1, \dots, n \quad j = 1, \dots, m \\ & y_j \in (0, 1) \quad j = 1, \dots, m \end{aligned} \quad (11.7)$$

Dopodiché si utilizzerà R per risolvere il problema. In questo caso, il vero problema è che finora abbiamo visto dei vincoli o delle funzioni obiettivo che erano ad indice unico; in questo caso la funzione obiettivo è una matrice e i vincoli si trovano ad essere su più indici assieme. Quello che dobbiamo fare è fare attenzione a linearizzare questi termini: dobbiamo mappare questa struttura su un array. E' l'unica difficoltà di questo esercizio.

Quindi il numero di variabili totali sarà pari a $n * m + m$ dove:

- $n * m$ sono le $x_{i,j}$;
- m sono le y_j ; (ovvero le m facilities)

All'interno del codice si ha che:

- `costs <- c(distCosts, building_costs$building_cost)` -> equivale ai coefficienti della funzione obiettivo dove il primo array è relativo ai costi delle distanze e il secondo array è relativo ai costi di costruzione degli hub;

Lo passeremo come parametro alla funzione `set.objfn` per creare la funzione obiettivo. Dopo tale funzione abbiamo la spiegazione di **come linearizzare una matrice**: in questo caso la matrice è linearizzata per riga. Inoltre, viene spiegato come mappare una certa posizione della matrice su questa riga. Il risultato finale che si otterrà, plottato, sarà:

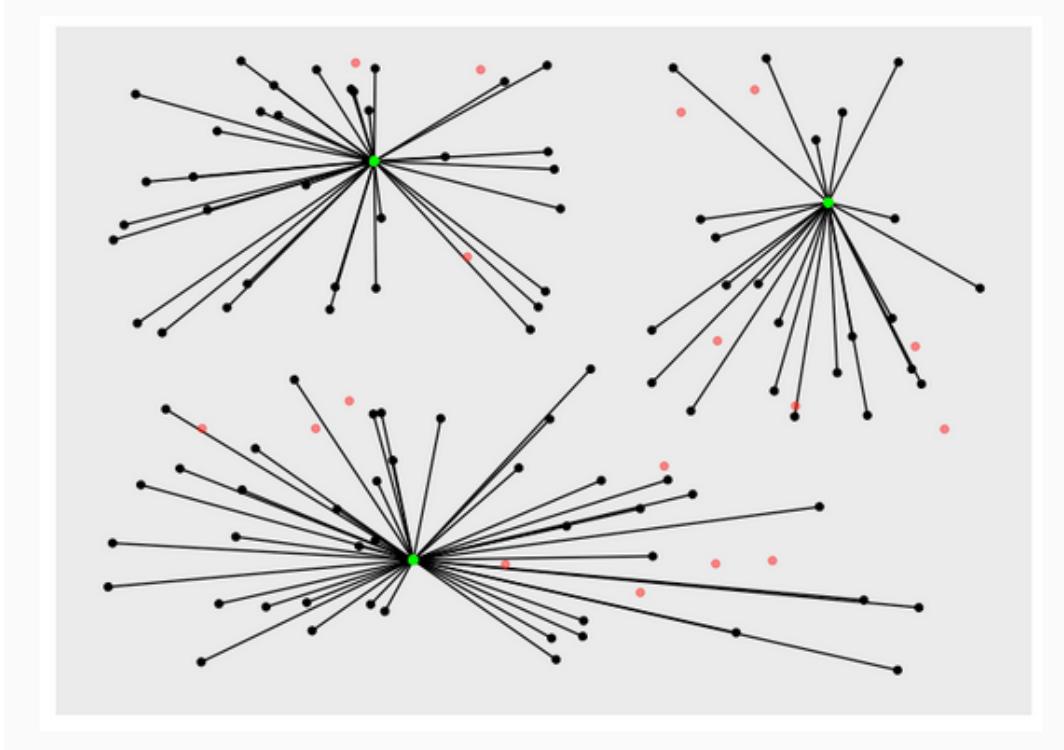


Figure 11.3: Plot hub finale

Soltanto tre posizioni sono state scelte e tutti i clienti sono stati assegnati ad una sola facility e tutte le altre facility non sono state scelte.

NOTA: LPSolve, il solver che usiamo per risolvere questo problema, è in grado di risolvere sia problemi di programmazione lineare continua, che intera che mista. Implementa sia il simplex che alcune varianti del branch-and-bound.

Estendiamo il problema imponendo che sia limitato il numero di facility da costruire. Partendo dal modello precedente, ovvero:

$$\begin{aligned}
 MIN : & \sum_{i=1}^n \sum_{j=1}^m distance(i, j) \times x_{i,j} + \sum_{j=1}^m building\ costs(j) \times y_j \\
 S.t. : & \sum_{j=1}^m x_{i,j} = 1 \quad i = 1, \dots, n \\
 & x_{i,j} \leq y_j \quad i = 1, \dots, n \quad j = 1, \dots, m \\
 & x_{i,j} \in (0, 1) \quad i = 1, \dots, n \quad j = 1, \dots, m \\
 & y_j \in (0, 1) \quad j = 1, \dots, m
 \end{aligned} \tag{11.8}$$

che vincolo devo aggiungere per fare in modo che il numero di facility da costruire sia **al massimo uguale a 2?** Dovremo sommare sulla y e imporre che questo valore sia minore/uguale di 2 in quanto y rappresenta la decisione di costruire o meno e se sommo su j mi verrà detto quante facility vogliamo costruire. Posso imporre che la mia soluzione non abbia più di due facility, ovvero:

$$\sum_{j=1}^m y_j \leq 2 \tag{11.9}$$

Il problema così più vincolato, essendo di minimizzazione, fa sì che il valore della funzione obiettivo sarà maggiore o uguale rispetto a quello della soluzione del problema originale. Ottenendo quindi:

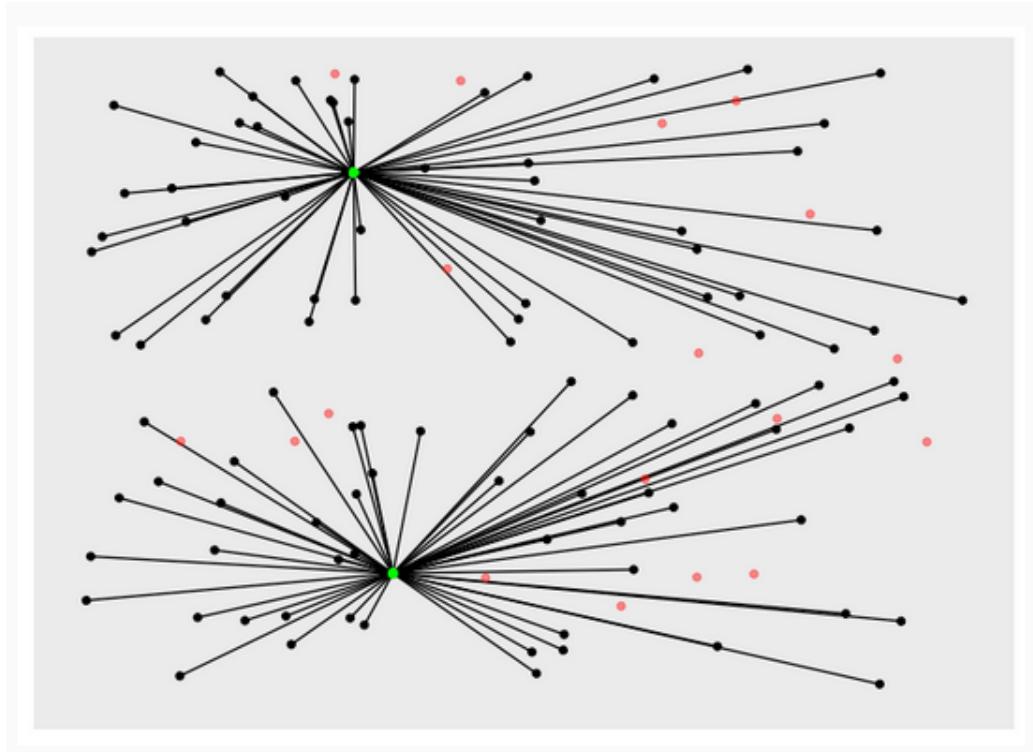


Figure 11.4: Plot hub finale 2

Successivamente si è provato a vedere cosa succederebbe se cambiassimo il metodo di calcolo della distanza che influisce sulla soluzione. In generale, si assume che la distanza sia calcolata tramite la distanza euclidea e che siano simmetriche: andare da A a B costi esattamente uguale ad andare da B ad A. In realtà, specie in città, ma anche fuori città, le distanze tra i punti possono essere meglio approssimati tenendo in considerazione distanze alternative: in **città** la distanza manhattan che è **la somma delle distanze rispetto all'asse x ed y** è una distanza che meglio approssima il percorso di un mezzo. Vediamo quindi qual è il risultato e l'assegnamento che viene fornito. Ciò che si ottiene sarà:

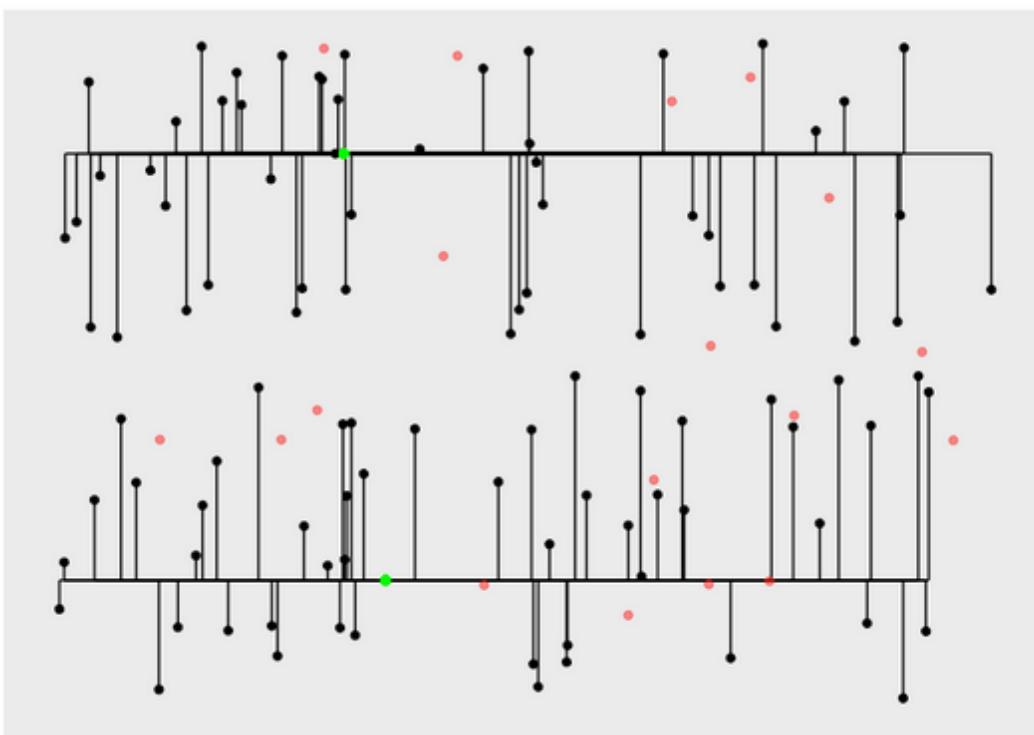


Figure 11.5: Plot hub finale 2

utilizzando la distanza Manhattan come distanza di disegno dei punti.

Ex. 9 - Network modelling - The Museum Problem

Guarderemo solo la parte di modellazione, mentre la parte di risoluzione con R è lasciata a noi da guardare.

L'idea alla base è la seguente: si ha un edificio in cui sono posizionate le seguenti stanze:

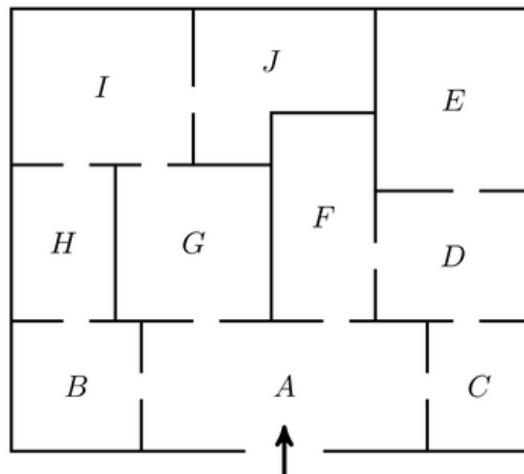
// The Museum Problem

A museum director must decide **how many security guards** should be employed to watch over a new wing of the building.

Budget cuts have forced him to station guards **only at the doors**, guarding two rooms at once.

Formulate a mathematical program to **minimize the number of guards** in order to **watch over all rooms** of the new wing.

The planimetry to consider is reported in the following figure.



/// Network Modelling

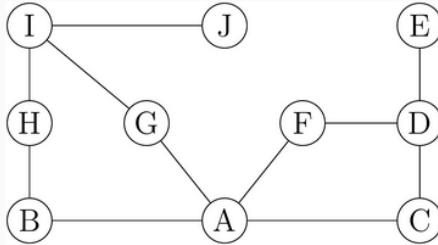
Figure 11.6: The Museum Problem

Le stanze sono collegate fra di loro da delle porte ed è necessario mettere una persona di guardia alle porte in modo tale da coprire tutte le stanze per dare una certa sicurezza. L'obiettivo è di ottenere il numero minimo di guardiani sulla porta o di porta attive per coprire l'intera area. Questo tipo di problemi si mappa bene al concetto di grafo. Possiamo immaginare ogni **stanza** come un **nodo** e ogni **porta** come un **arco**. Il risultato sarà:

/// Network Modelling

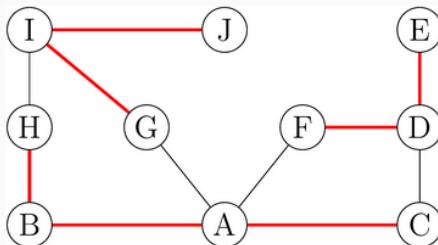
The problem can be written in terms of a graph problem easily: we represent each room by a vertex $v \in V$ of an undirected graph $G = (V, E)$ where E is the set of doors.

We create an edge between two vertexes if there is a door joining the corresponding rooms, this way, edges (doors) represent the potential position of a guards.



We want to find the **smallest subset $F \subseteq E$ of edges covering** all vertexes, i.e. such that for all $v \in V$ there is an edge e in the set F such that v is one extreme of e .

For example a possible solution could be put guards in the doors $F = \{IJ, IG, BH, ABAC, FD, DE\}$ as depicted in the following figure:



Even if in this case the solution is easy to find (take a moment to find it!) the problem could be difficult to solve if the number of rooms is large.

Figure 11.7: The Museum Problem Solution

Vogliamo trovare il **set di archi più piccolo** (minore/uguale) dell'insieme totale degli archi che copra tutti i nodi. Sceglierò quegli archi in modo tale che tutti i nodi abbiano almeno un arco entrante. Sarà quindi possibile scartare gli archi tra I-H e tra G-A e tra F-A e tra D-C. Scegliere tutti gli archi sarebbe una soluzione ammissibile per il nostro problema ma non sarebbe ottimale.

Proviamo a capire quale è l'**insieme delle variabili decisionali** che bisogna utilizzare per formulare il problema:

- Devo dire: arco attivo/arco non attivo. Definiamo delle variabili $x_{i,j}$ dove i e j rappresentano entrambi dei nodi e queste variabili devono essere vincolate ad essere 0 o 1.
- Qual è la funzione obiettivo? Dobbiamo minimizzare la somma di queste variabili $x_{i,j}$, ovvero avere il numero minimo di archi attivi;
- I vincoli: noi vogliamo che tutti i nodi siano coperti, ovvero che sia presente almeno un arco uscente/entrante da ogni nodo. Essendo i e j dei nodi, se sommo su i o su j è uguale e devo imporre che siano maggiori/uguali di 1: deve essere almeno un arco uscente e almeno un arco entrante.

In definitiva:

// Integer Linear Programming Formulation

Thus, we will formulate the problem as an ILP:

1. **Decision variables:** For each edge $(i, j) \in E(G)$ we create a binary (decision) variable $x_{ij} \in \{0, 1\}$ that will assume value 1 if and only if there is a guard on the door represented by edge (i, j) and 0 otherwise.
2. **Objective function:** In this way the total number of guards will be the sum over all edges of the variables x_{ij} , that is $\sum_{(i,j) \in E} x_{ij}$ and it corresponds to the quantity that we aim to minimize.
3. **Constraints:** Finally, we must ensure that every room is guarded. In other words, we must guarantee that for each room (vertex) i there is **at least one guard** on one door (active edge) that communicate room i to the other rooms. In terms of our decision variables this constraint for the room i can be expressed as:

$$\sum_{j \in V : (i,j) \in E} x_{ij} \geq 1 \quad \forall i \in V$$

Putting all together, we get the following formulation:

$$\begin{aligned} \min \quad & \sum_{(i,j) \in E} x_{ij} \\ \text{subject to} \quad & \sum_{j \in V : (i,j) \in E} x_{ij} \geq 1 \quad \forall i \in V \quad (\text{vertex covering}) \\ & x_{ij} \in \{0, 1\} \quad (\text{binary}) \end{aligned}$$

In our example:

$$\begin{array}{lllll} \min & AB + AC + AG + AF + BH + CD + DF + DE + GI + IH + IJ \\ A : & AB + AC + AG + AF & & & \geq 1 \\ B : & AB & + BH & & \geq 1 \\ C : & AC & + CD & & \geq 1 \\ D : & & CD + DF + DE & & \geq 1 \\ E : & & & DE & \geq 1 \\ F : & AF & + DF & & \geq 1 \\ G : & AG & & + GI & \geq 1 \\ H : & & BH & + IH & \geq 1 \\ I : & & & GI + IH + IJ & \geq 1 \\ J : & & & & IJ \geq 1 \\ \text{subject to} & AB, AC, AG, AF, BH, CD, DF, DE, GI, IH, IJ \in \{0, 1\} \end{array}$$

Figure 11.8: The Museum Problem Solution 1

we now rename the variables and put the model in matricial form:

$$\begin{aligned}
 & \min \quad (1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1) \bar{y}^t \\
 & \text{s.t.} \quad \left[\begin{array}{ccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right] \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \end{pmatrix} \geq \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \\
 & \bar{y} \in \{0, 1\}^{11}
 \end{aligned}$$

It is important to note that, compared to a "classical" ILP formulation, we exploit the known graph structure (V and E) to reduce the number of variables and constraints to make the model simpler and easier to solve. In fact, notice that the constraints matrix is practically an adjacency matrix.

Figure 11.9: The Museum Problem Solution 2

Questo si chiama **problema su grafo/rete** perché non tutte le $x_{i,j}$ possono essere considerate, in quanto tra alcuni nodi non esiste una porta di collegamento diretto. Quindi posso escludere a priori un certo numero di variabili. Quindi questo problema che partiva con n^2 variabili (avendo n nodi), ha in realtà molte meno variabili esattamente pari al numero di archi del mio modello, in questo caso il numero di archi è pari a 11 e il modello si può semplificare passando dalla variabile $x_{i,j}$ ad una variabile y_i dove questa volta il set che consideriamo è il set degli archi. La matrice di poco sopra è una matrice sparsa cosiddetta di (quasi) **adiacenza**: *quasi* perché manca la diagonale principale che rappresenterebbe la relazione di un nodo con sè stesso (in questo caso non serve). Avremo quindi un numero di variabili molto ridotto e la matrice rappresenta in maniera compatta la struttura del grafo, ovvero in maniera sparsa. La matrice rappresenta le stanze su righe e colonne e ogni volta che la stanza è in comune ho un 1 (è una sorta di mappa). In questo modo non mi serve considerare tutte le variabili ma solo quelle che sono utili alla risoluzione del mio problema.

Chapter 12

LEZIONE 10 - (19/04/2021)

In questa lezione vediamo dei problemi di modellazione particolare. Molti dei problemi di modellizzazione vengono formulati a partire da una rappresentazione su rete/grafi: questo perché la rappresentazione grafica aiuta nella formulazione e nelle interpretazioni del modello, ma anche perché il grafo ha una struttura particolare e quindi i problemi formulati su grafo possono avere delle strutture che possono essere sfruttate per progettare degli algoritmi efficienti. Abbiamo visto che i problemi su grafo sono generalmente problemi **discreti**, quindi di programmazione lineare intera/mista e abbiamo visto quanto sia difficile risolvere problemi di programmazione intera, vista la non banale complessità se il numero di variabili è molto elevato. Sfruttando la struttura del grafo sono stati creati algoritmi ad hoc. Nello specifico andremo a vedere la modellazione di alcuni algoritmi senza entrare nel dettaglio dei singoli casi.

In questa lezione ci si focalizzerà su diversi problemi come, ad esempio:

- Transshipment Problems; (di trasbordo)
- Shortest Path Problems; (di cammino minimo su grafo tra un nodo e un altro)
- Maximal Flow Problems; (problemi di massimo flusso che può attraversare un arco della rete)
- Transportation/Assignment Problems;
- Generalized Network Flow Problems;
- The Minimum Spanning Tree Problems;

Quali sono le **caratteristiche principali** dei problemi di flusso?

- i problemi su rete possono essere rappresentati da **grafi** composti da **nodi** connessi da **archi**;
- ci sono generalmente tre tipi di nodi:
 - **Nodi Supply**: dove viene prodotto qualcosa o sono l'origine di qualcosa;
 - **Nodi Demand**: dove si ha una richiesta di qualcosa;
 - **Nodi Transshipment**: che non sono né nodi Supply né Demand. Sono nodi di attraversamento: tutto il flusso che entra deve anche uscire;
- Per modellare questi problemi ci sono vari modi: noi useremo la convenzione per cui i numeri negativi rappresentano le offerte dei Nodi Supply, mentre i nodi positivi rappresentano i Nodi Demand;

Quindi, vediamoli uno per uno:

1. **Transshipment Problems; (di trasporto)**:

A Transshipment Problem: The Bavarian Motor Company

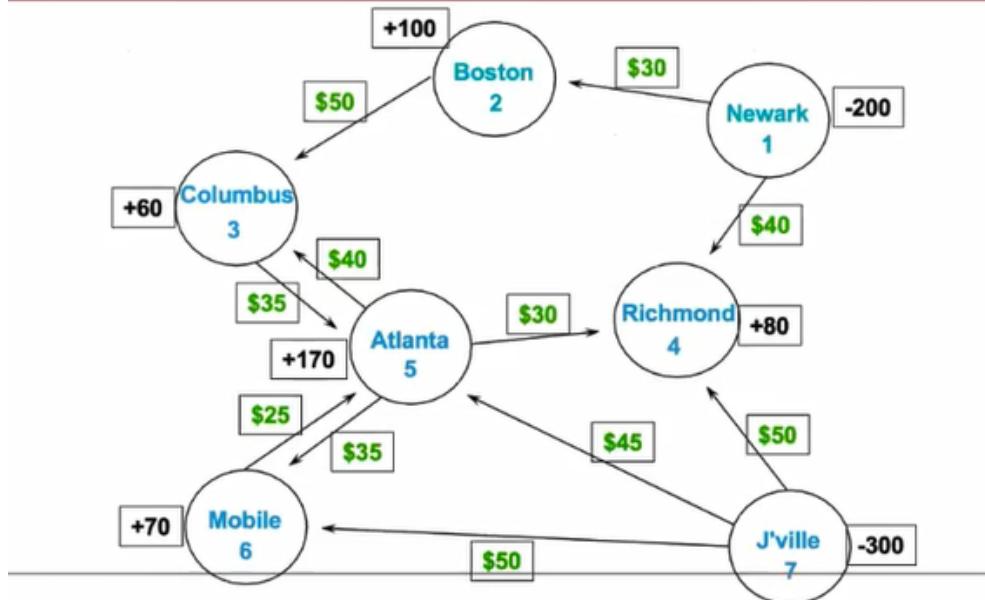


Figure 12.1: Transshipment Problem 1

Qual è la differenza fra i problemi di **transshipment** (di trasporto) e i problemi di **trasporto**? Che non hanno solo nodi Supply e nodi Demand, ovvero il problema non l'avere solo nodi in cui produco qualcosa e una serie di nodi in cui richiedo i prodotti e questi nodi sono connessi tra di loro. Sono presenti appunto questi nodi di transshipment che sono dei nodi che non sono né Supply né Demand che devo attraversare per arrivare dall'origine alle destinazioni.

Supponiamo di avere un problema: si hanno autoveicoli prodotti in alcuni stabilimenti come *Newark* e *Jacksonville* (i numeri negativi rappresentano le Supply, quindi nel nodo I e nel nodo VII si ha una produzione) e si ha una richiesta in altre città (*Boston*, *Columbus*, *Atlanta*, *Richmond*, *Mobile*).

Sui nodi abbiamo l'informazione che il nodo sia un nodo dove si **origina il flusso**, oppure un nodo che **assorbe il flusso**. I numeri in verde rappresentano i **costi unitari di traversamento** dell'arco.

La nostra decisione in questi problemi deve essere come riuscire spedire i prodotti dai nodi sorgente ai nodi destinazione minimizzando il costo di spedizione. Le decisioni che dobbiamo prendere sono: **quanto spedire da un nodo ad un altro? Ovvero, qual è il flusso di prodotti che attraversa un arco?** Quindi il numero di variabili sarà uguale al numero di archi del mio grafo. Mentre i vincoli saranno definiti sui nodi in quanto il flusso che esce da Newark dovrà essere non più grande di 200 perché quella è la produzione; quanto esce da Jacksonville non potrà essere più grande di 300. Quello che arriva a Richmond dovrà essere almeno di 80. Quindi sui nodi vengono definiti i vincoli.

Per cui:

- Per ogni arco della rete ci sarà una variabile di decisione X_{ij} che sarà la quantità spedita dal nodo i al nodo j ; per esempio, x_{12} sono il numero di macchine spedite dal nodo 1 al nodo 2; x_{56} sono il numero di macchine spedite dal nodo 5 al nodo 6 etc.

Quindi **il numero di archi determina il numero delle variabili**.

La funzione obiettivo sarà la **minimizzazione del costo** e quindi del costo totale di spedizione:

$$MIN : 30X_{12} + 40X_{14} + 50X_{23} + 35X_{35} + 40X_{53} + 30X_{54} + 35X_{56} + 25X_{65} + 50X_{74} + 45X_{75} + 50X_{76} \quad (12.1)$$

e poi ci sono i **vincoli** che riguardano i nodi: ad ogni nodo devo associare un vincolo. Ci possono essere diverse situazioni per questi problemi di minimizzazione del flusso su una rete. Posso avere i seguenti casi:

Constraints for Network Flow Problems: The Balance-of-Flow Rules

For Minimum Cost Network Flow Problems Where:	Apply This Balance-of-Flow Rule At Each Node:
Total Supply > Total Demand	Inflow-Outflow \geq Supply or Demand
Total Supply \downarrow < Total Demand	Inflow-Outflow \leq Supply or Demand
Total Supply = Total Demand	Inflow-Outflow = Supply or Demand

Figure 12.2: Vincoli

Ovvero, partendo dall'alto a sinistra:

- Caso 1: l'offerta totale è maggiore della domanda: avrà che il flusso entrante in un arco potrà essere maggiore/uguale della Supply (inteso come numero negativo) o della Demand;
- Caso 2: l'offerta totale è minore della domanda: avrà che il flusso entrante in un arco potrà essere minore/uguale della Supply (inteso come numero negativo) o della Demand;
- Caso 3: l'offerta totale e la domanda totale sono esattamente uguali: avrà che il flusso entrante in un arco sarà uguale della Supply (inteso come numero negativo) o della Demand;

Guardiamo allo specifico problema:

- Total Supply = 500 macchine;
- Total Demand = 480 macchine;

Quindi **Supply \geq Demand** (in quanto si hanno $200 + 300$ macchine).

Per ogni nodo si inserirà un vincolo in cui **flusso in ingresso - flusso in uscita \geq Supply (negativa) o Demand**.

Ad esempio, vediamo il vincolo sul nodo 1:

$$-X_{12} - X_{14} \geq -200 \quad (\text{non c'è inflow per il nodo 1}) \quad (12.2)$$

In questo caso il flusso in ingresso è 0. Il flusso in uscita è dettato da due variabili: X_{12} e X_{14} . La somma del flusso in uscita deve essere maggiore/uguale della Supply, cioè dell'offerta che c'è in quel nodo. Equivalentemente a:

$$X_{12} + X_{14} \leq 200 \quad (12.3)$$

L'inflow è pari a 0 a cui si sottrae l'outflow che è $-X_{12} - X_{14}$ dovrà essere maggiore/uguale a -200 (ovvero, quello che esce da quel nodo non potrà essere più di 200). Analogamente per gli altri nodi, ovvero:

Defining the Constraints

- Flow constraints

$$\begin{aligned}
 -X_{12} - X_{14} &\geq -200 && \} \text{ node 1} \\
 +X_{12} - X_{23} &\geq +100 && \} \text{ node 2} \\
 +X_{23} + X_{53} - X_{35} &\geq +60 && \} \text{ node 3} \\
 +X_{14} + X_{54} + X_{74} &\geq +80 && \} \text{ node 4} \\
 +X_{35} + X_{65} + X_{75} - X_{53} - X_{54} - X_{56} &\geq +170 && \} \text{ node 5} \\
 +X_{56} + X_{76} - X_{65} &\geq +70 && \} \text{ node 6} \\
 -X_{74} - X_{75} - X_{76} &\geq -300 && \} \text{ node 7}
 \end{aligned}$$

- Nonnegativity conditions

$$X_{ij} \geq 0 \text{ for all } ij$$

Figure 12.3: Definizione dei vincoli

ATTENZIONE: il nodo 2 è un nodo Demand (e non Supply) per cui avrà il maggiore/uguale e così via per gli altri nodi. In generale: se **inflow** allora positivo; se **outflow** allora negativo.

Dopodiché abbiamo le condizioni di non negatività. Altri vincoli che ci possono essere potrebbero essere relativi alla capacità degli archi: può essere che lungo una tratta ci siano dei limiti superiore rispetto alla quantità di flusso che può attraversarli. In questo caso ci saranno quindi oltre i vincoli di non negatività anche degli **upper bound** sulle variabili X_{ij} .

Una cosa importante che possiamo notare è che i coefficienti dei vincoli se facciamo in modo che ogni variabile sia rappresentata da una colonna e ogni vincolo da un nodo, abbiamo che questa matrice ha una struttura particolare: ha tutti elementi che sono $+/-1$ e ogni colonna avrà esattamente un $+/-1$ ovvero un'origine e una destinazione. Se proviamo a risolvere questo problema, la soluzione è la seguente:

Optimal Solution to the BMC Problem

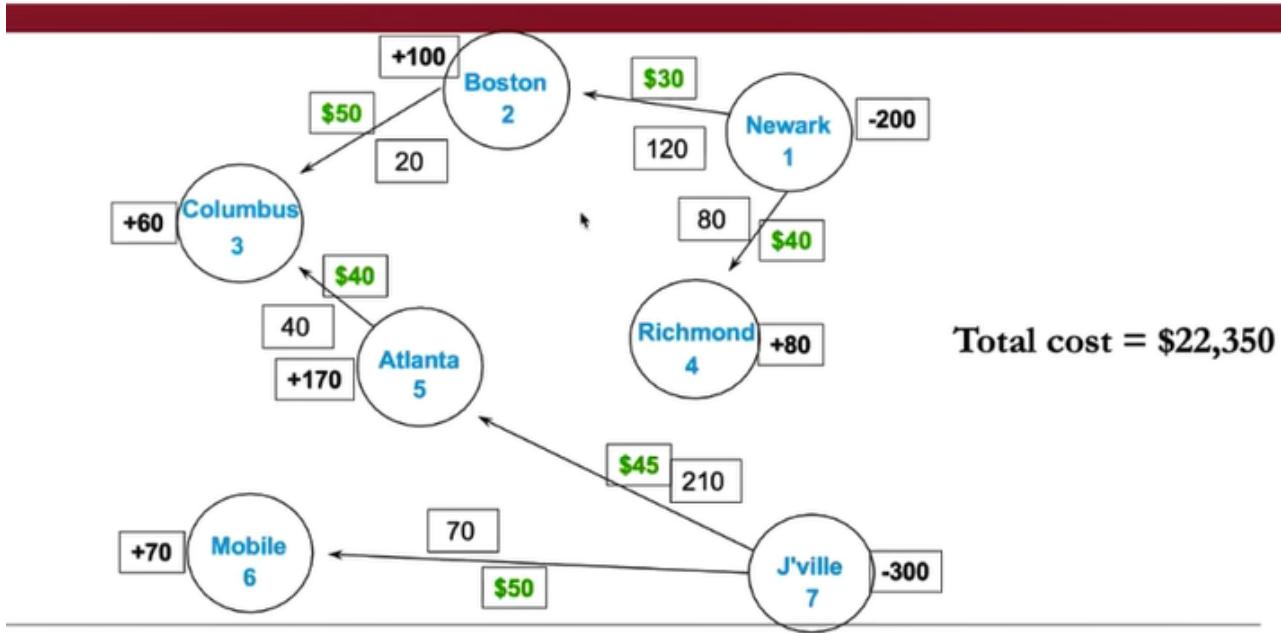


Figure 12.4: Soluzione ottima

Si nota come la soluzione sia a numeri interi che non è un caso come quello che avevamo visto nel primo esempio del corso relativo alle Hot Tubs: in questo caso, quando i coefficienti dei termini noti sono interi allora la soluzione di un problema di trasporto sarà a numeri interi. Quindi questa struttura particolare della matrice (definita **unimodularità della matrice**) mi garantisce che il problema ha una soluzione a numeri interi. Posso usare l'algoritmo del simplex e sono sicuro che la mia soluzione avrà numeri interi. Oltre a ciò ci sono algoritmi specializzati per risolvere tali problemi. Nella matrice di incidenza le **righe sono i nodi** e le **colonne sono gli archi** e per sapere da dove parte dove arriva l'arco metto un +1 o un -1 per origine o destinazione o viceversa (a seconda della convenzione scelta). I vincoli definiti quindi se esplicitiamo la matrice dei vincoli con le variabili sulle colonne e i vincoli sulle righe e mettiamo i vari coefficienti nel posto giusto, troviamo la matrice di incidenza del grafo: la matrice dei vincoli equivale alla matrice di incidenza del grafo.

2. Shortest Path Problems; (di cammino minimo su grafo tra un nodo e un altro):

Altro problema tipico è quello del **cammino minimo** (quello dei navigatori GPS) in base a dei criteri/vincoli (il cammino minimo tra due nodi). Questo problema quindi consiste nel determinare il **percorso minimo tra due nodi di una rete**. Un derivato di questo problema è il problema del **vehicle routing** (esempio: Amazon che deve fare in modo che la merce arrivi a destinazione velocemente; riguarda tutti i problemi di trasporti da magazzini e per le consegne; tutto ciò può essere rappresentato come un tale problema). Si tratta di un caso particolare del problema del *transshipment* precedente in cui:

- Si ha un solo nodo Supply dove l'offerta è -1;
- Si ha un solo nodo Demand dove la domanda è +1;
- Tutti gli altri nodi hanno Supply/Demand di +0.

Supponiamo di avere il seguente problema:

The American Car Association

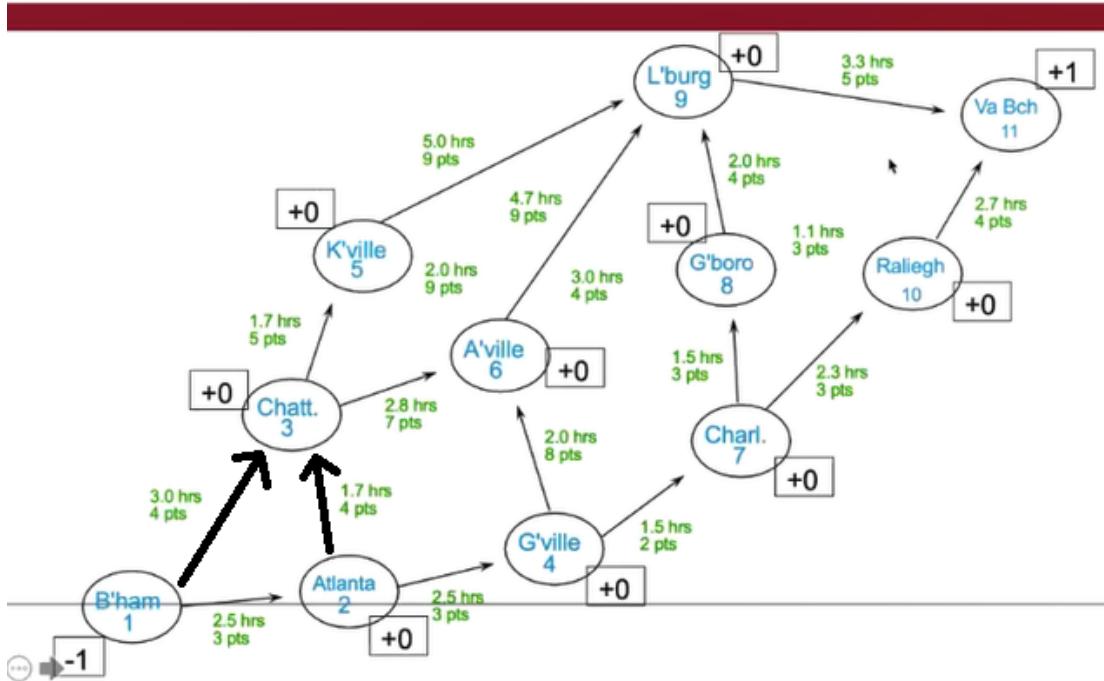


Figure 12.5: Shortest Path

Dove devo trasportare una unità di prodotto dal nodo I in basso a sinistra al nodo XI in alto a destra, dove il percorso passa per le città intermedie. Si ha che il **nodo supply** è il primo con -1 e il nodo demand XI è $+1$ in alto a destra. Tutti gli altri nodi sono **nodi transshipment**. Per andare dal nodo I al nodo XI si hanno diversi percorsi possibili e ogni arco è pesato con due possibili pesi e unità di misura:

- Ore: il tempo che ci si impiega per attraversare l'arco;
- Bellezza del paesaggio: si assegna un punteggio a seconda della bellezza del percorso;

L'obiettivo può essere duplice:

- Trovare il percorso minimo rispetto al tempo (minimizzare il tempo di viaggio);
- Trovare il percorso più bello rispetto al paesaggio (massimizzare il rating del paesaggio);
- Trovare una combinazione tra le due, ma questo vorrebbe dire aggiungere un parametro nella funzione obiettivo che possa rappresentare una combinazione lineare tra queste due funzioni obiettivo e questo parametro andrebbe ad avere il significato del peso associato ad ogni obiettivo (posso dare più peso alla velocità e meno al paesaggio o viceversa);

Noi vedremo il problema con una sola funzione obiettivo e non con una funzione multiobiettivo. Anche in questo caso le variabili sono associate agli archi che sono delle variabili 0/1: quindi l'arco lo seleziono oppure non lo seleziono nel mio percorso. Quindi selezionare un percorso significa selezionare quella serie di archi che costituiscono quel percorso. La formulazione è del tutto analoga a quella vista in precedenza. Prendendo in considerazione il tempo si avrà:

Solving the Problem

$$\text{MIN: } +2.5X_{12} + 3X_{13} + 1.7X_{23} + 2.5X_{24} + 1.7X_{35} + 2.8X_{36} + 2X_{46} + 1.5X_{47} + 2X_{56} + 5X_{59} \\ + 3X_{68} + 4.7X_{69} + 1.5X_{78} + 2.3X_{7,10} + 2X_{89} + 1.1X_{8,10} + 3.3X_{9,11} + 2.7X_{10,11}$$

Subject to:

$-X_{12} - X_{13}$	$= -1$	flow constraint for node 1
$+X_{12} - X_{23} - X_{24}$	$= 0$	flow constraint for node 2
$+X_{13} + X_{23} - X_{35} - X_{36}$	$= 0$	flow constraint for node 3
$+X_{24} - X_{46} - X_{47}$	$= 0$	flow constraint for node 4
$+X_{35} - X_{56} - X_{59}$	$= 0$	flow constraint for node 5
$+X_{36} + X_{46} + X_{56} - X_{68} - X_{69}$	$= 0$	flow constraint for node 6
$+X_{47} - X_{78} - X_{7,10}$	$= 0$	flow constraint for node 7
$+X_{68} + X_{78} - X_{89} - X_{8,10}$	$= 0$	flow constraint for node 8
$+X_{59} + X_{69} + X_{89} - X_{9,11}$	$= 0$	flow constraint for node 9
$+X_{7,10} + X_{8,10} - X_{10,11}$	$= 0$	flow constraint for node 10
$+X_{9,11} + X_{10,11}$	$= +1$	flow constraint for node 11
$X_{ij} \geq 0$ for all i and j		nonnegativity conditions

Figure 12.6: Shortest problem solution

La formulazione è praticamente uguale a quella di prima con l'unica differenza che i termini noti sono tutti zero tranne l'origine e la destinazione. L'interpretazione dei vincoli è la stessa (con *inflow* e *outflow*). La soluzione sarà quindi:

Solving the Problem

Select Route?	From	To	Driving Time	Scenic Rating
1.0	1 Birmingham	2 Atlanta	2.5	3
0.0	1 Birmingham	3 Chattanooga	3.0	4
0.0	2 Atlanta	3 Chattanooga	1.7	4
1.0	2 Atlanta	4 Greenville	2.5	3
0.0	3 Chattanooga	5 Knoxville	1.7	5
0.0	3 Chattanooga	6 Asheville	2.8	7
0.0	4 Greenville	6 Asheville	2.0	8
1.0	4 Greenville	7 Charlotte	1.5	2
0.0	5 Knoxville	6 Asheville	2.0	9
0.0	5 Knoxville	9 Lynchburg	5.0	9
0.0	6 Asheville	8 Greensboro	3.0	4
0.0	6 Asheville	9 Lynchburg	4.7	9
0.0	7 Charlotte	8 Greensboro	1.5	3
1.0	7 Charlotte	10 Raleigh	2.3	3
0.0	8 Greensboro	9 Lynchburg	2.0	4
0.0	8 Greensboro	10 Raleigh	1.1	3
0.0	9 Lynchburg	11 Virginia Beach	3.3	5
1.0	10 Raleigh	11 Virginia Beach	2.7	4
			Total	11.5
				15

Figure 12.7: Shortest problem solution 2

Il tempo totale sarà 11.5 e il rating del paesaggio pari a 15.

Scgliendo di massimizzare invece la bellezza del paesaggio si avrà:

Solving the Problem

Select Route?	From	To	Driving Time	Scenic Rating
1.0	1 Birmingham	2 Atlanta	2.5	3
0.0	1 Birmingham	3 Chattanooga	3.0	4
1.0	2 Atlanta	3 Chattanooga	1.7	4
0.0	2 Atlanta	4 Greenville	2.5	3
1.0	3 Chattanooga	5 Knoxville	1.7	5
0.0	3 Chattanooga	6 Asheville	2.8	7
0.0	4 Greenville	6 Asheville	2.0	8
0.0	4 Greenville	7 Charlotte	1.5	2
1.0	5 Knoxville	6 Asheville	2.0	9
0.0	5 Knoxville	9 Lynchburg	5.0	9
0.0	6 Asheville	8 Greensboro	3.0	4
1.0	6 Asheville	9 Lynchburg	4.7	9
0.0	7 Charlotte	8 Greensboro	1.5	3
0.0	7 Charlotte	10 Raleigh	2.3	3
0.0	8 Greensboro	9 Lynchburg	2.0	4
0.0	8 Greensboro	10 Raleigh	1.1	3
1.0	9 Lynchburg	11 Virginia Beach	3.3	5
0.0	10 Raleigh	11 Virginia Beach	2.7	4
Total			15.9	35

Figure 12.8: Shortest problem solution 3

3. The Equipment Replacement Problem:

Un altro problema è quello della sostituzione di strumentazione. Cosa c'entra con i grafi? Questo problema può essere definito tramite un cammino su grafo. Questo significa che diversi algoritmi possono essere utilizzati per risolvere problemi differenti. Un'abilità dei ricercatori operativi è quella di mappare un modello di una qualsiasi applicazione che hanno su un problema di ottimizzazione su grafo, in modo tale da utilizzare algoritmi più efficienti.

Il problema è quello di determinare quando sostituire delle strumentazioni, ricadendo all'interno della manutenzione predittiva. Vediamo un esempio.

La Compu-Train Company fornisce del training tramite software e i computer che sono usati per questo training devono essere cambiati ogni due anni per garantirne il funzionamento. E' possibile scegliere tra due contratti di leasing, ciascuno dei quali richiede 62000 inizialmente e poi:

Contratto 1:

- Aumenta il suo prezzo del 6% per anno;
- Restituisce il 60% del costo del vecchio equipaggiamento dopo 1 anno;
- Oppure restituisce il 15% del costo del vecchio equipaggiamento dopo 2 anni;

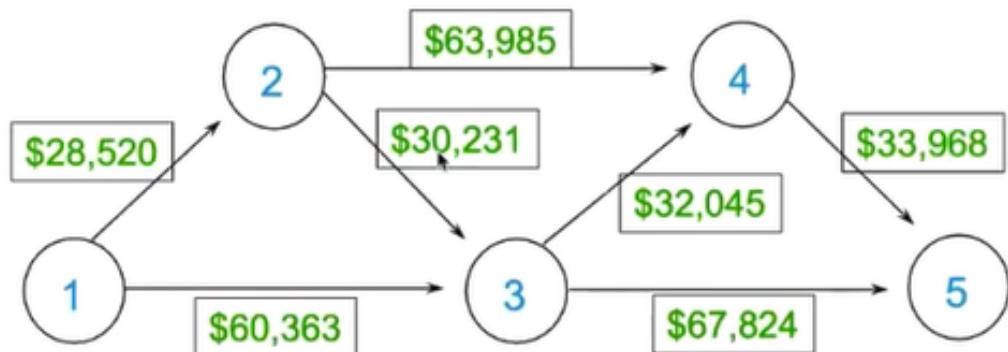
Contratto 2:

- Aumenta il suo prezzo del 2% per anno;

- Restituisce il 30% del costo del vecchio equipaggiamento dopo 1 anno;
- Oppure restituisce il 10% del costo del vecchio equipaggiamento dopo 2 anni;

Posso quindi rappresentare tale problema su grafo, ovvero:

Network for Contract 1



Cost of trading after 1 year: $1.06 * \$62,000 - 0.6 * \$62,000 = \$28,520$

Cost of trading after 2 years: $1.06^2 * \$62,000 - 0.15 * \$62,000 = \$60,363$

etc, etc....

Figure 12.9: newtork contract 1

Quindi rimpiazzare tutti i computer dopo 1 anno ha il costo di 28520.

Se lo rimpiazzo dopo 2 anni, allora avrò che il costo sarà di 60363. (i numeri nei pallini sono gli anni)

Ciò che si deve fare è trovare il cammino minimo tra il nodo 1 e il nodo 5. Le possibili soluzioni saranno (a seconda che sostituisca ogni anno oppure ogni due anni):

Solving the Problem

The figure consists of two tables side-by-side, each representing a solution to a problem involving a set of edges and their costs.

Select	From	To	Cost
1.0	1	2	\$28,520
0.0	1	3	\$60,363
1.0	2	3	\$30,231
0.0	2	4	\$63,985
1.0	3	4	\$32,045
0.0	3	5	\$67,824
1.0	4	5	\$33,968

Select	From	To	Cost
0.0	1	2	\$44,640
1.0	1	3	\$58,305
0.0	2	3	\$45,533
0.0	2	4	\$59,471
0.0	3	4	\$46,443
1.0	3	5	\$60,660
0.0	4	5	\$47,372

Total Cost \$124,764

Total Cost \$118,965

Figure 12.10: contract solution

Tutto ciò per mostrare che il problema del cammino minimo su grafo è usato per risolvere non solo problemi di dominio che riguardano un percorso su un grafo in termini di distanza, cioè di viaggio. Ma posso anche utilizzare questi modelli per risolvere problemi diversi, come quello della manutenzione predittiva.

4. Transportation & Assignment Problems:

Vediamo ora i problemi di trasporto e assegnamento: in questo caso si tratta di problemi di flussi su reti in cui non ci sono i nodi di transhipment, ma spedisco direttamente da una nodo ad una destinazione.

Perché si mettono assieme trasporto e assegnamento? Perché hanno due strutture molto simili: se devo trasportare degli agrumi dal campo agli stabilimenti, ho degli archi diretti dai campi agricoli agli stabilimenti per il processamento. Devo in qualche modo assegnare l'arco che mi determina la quantità di agrumi trasportate da un nodo sorgente ad un nodo destinazione. Così come transshipment e il cammino minimo (caso particolare del transshipment) l'assignment è un caso particolare del problema del trasporto dove devo assegnare (quindi selezionare/non selezionare) un arco e devo assegnare ogni nodo origine ad un nodo destinazione. Questo significa che la supply sarà 1 per ogni nodo e la capacity sarà 1 per ogni nodo. Vengono quindi presentati assieme visto che l'assignment è un caso particolare del problema del trasporto.

Transportation & Assignment Problems

- Some network flow problems don't have trans-shipment nodes; only supply and demand nodes.

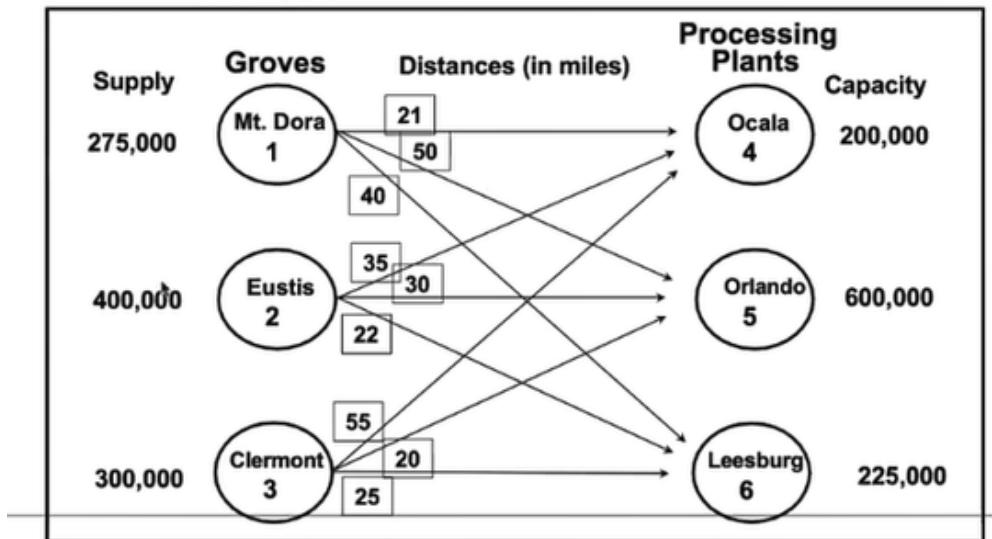


Figure 12.11: Transportation and Assignment

In questo caso abbiamo che l'offerta nei vari nodi sorgente sarà pari a 275000 per il nodo I, 400000 per il nodo II, 300000 per il nodo III e la capacità di processamento negli stabilimenti sarà 200000 per il nodo IV, 600000 per il nodo V, 225000 per il nodo VI. Bisogna quindi decidere quanto trasportare da ogni campo ad ogni stabilimento e i numeri sono i **costi di trasporto unitari**.

Le variabili sono **associate agli archi** e si avranno:

Defining the Decision Variables

$$X_{ij} = \# \text{ of bushels shipped from node } i \text{ to node } j$$

Specifically, the nine decision variables are:

- X_{14} = # of bushels shipped from Mt. Dora (node 1) to Ocala (node 4)
- X_{15} = # of bushels shipped from Mt. Dora (node 1) to Orlando (node 5)
- X_{16} = # of bushels shipped from Mt. Dora (node 1) to Leesburg (node 6)
- X_{24} = # of bushels shipped from Eustis (node 2) to Ocala (node 4)
- X_{25} = # of bushels shipped from Eustis (node 2) to Orlando (node 5)
- X_{26} = # of bushels shipped from Eustis (node 2) to Leesburg (node 6)
- X_{34} = # of bushels shipped from Clermont (node 3) to Ocala (node 4)
- X_{35} = # of bushels shipped from Clermont (node 3) to Orlando (node 5)
- X_{36} = # of bushels shipped from Clermont (node 3) to Leesburg (node 6)

Figure 12.12: Transportation and Assignment Variables

e la minimizzazione della funzione obiettivo sarà simile a quella del problema del precedente problema, ovvero minimizzare il costo del trasporto.

$$MIN : 21X_{14} + 50X_{15} + 40X_{16} + 35X_{24} + 30X_{25} + 22X_{26} + 55X_{34} + 20X_{35} + 25X_{36} \quad (12.4)$$

Però in questi problemi abbiamo **due insiemi di vincoli**:

Vincoli sull'offerta: (quello che produce il sito del nodo 1 è uguale a 275000, etc.)

$$X_{14} + X_{15} + X_{16} \leq 275000 \quad Mt.DoraX_{24} + X_{25} + X_{26} \leq 400000 \quad EustisX_{34} + X_{35} + X_{36} \leq 300000 \quad Clermont \quad (12.5)$$

Vincoli della domanda: (che mi dicono che quelle che entra nel nodo 4 deve essere minore/uguale di 200000, cioè non posso processare più di 200000 unità di agrumi; e così via)

$$X_{14} + X_{24} + X_{34} \leq 200000 \quad OcalaX_{15} + X_{25} + X_{35} \leq 600000 \quad OrlandoX_{16} + X_{26} + X_{36} \leq 225000 \quad Leesburg \quad (12.6)$$

Non-negativity conditions:

$$X_{ij} \geq \forall i \text{ and } j \quad (12.7)$$

In generale, sarebbe corretto che la Supply fosse uguale alla Demand. Se questo non succede o si usano i vincoli di disuguaglianza, oppure si può aggiungere un nodo fittizio laddove ho bisogno di far assorbire la quantità. Se, ad esempio, la Supply è maggiore della capacity io posso mettere un nodo destinazione fittizio nel quale far confluire tutto quello che avanza della Supply; se la Supply è minore della Demand allora posso mettere un nodo fittizio di Supply che va a sopprimere alla mancanza di produzione e se nella soluzione ottimale io assegno una quantità che viene da questo nodo è come se non soddisfassi una certa domanda. E' ovvio che in questo caso la richiesta non deve essere assolutamente di soddisfare la domanda perché se la produzione totale è minore della domanda totale, il problema non avrebbe soluzioni altrimenti. Però nel caso in cui si cerca di acquisire tutto quello che posso, allora ci si accontenta anche di una domanda parzialmente soddisfatta e per rappresentarlo posso aggiungere un nodo fittizio.

5. Generalized Network Flow Problems:

Nel caso in cui ci siano delle perdite o guadagni di flusso lungo gli archi dobbiamo modellare il problema andando ad aggiungere un fattore moltiplicativo che rappresenti questa cosa.

SI CONTINUA NELLA PROSSIMA LEZIONE.

Chapter 13

LEZIONE 11 - Practical Session Ex. 10 - Network Modelling (21/04/2021)

Con la lezione di oggi completiamo quanto visto nella precedente esercitazione. Vedremo una serie di problemi che si possono modellare attraverso sia modelli lineri continui che modelli lineari discreti. Se si fa uso del concetto di grafo presentano un numero minore di variabili perché le variabili si associeranno agli archi o ai nodi del grafico. Tutte quelle possibili coppie nodi-nodi che non esisterebbero nel grafo specifico non andranno considerati.

Problema del domino: una tessera del domino è composta da due quadrati adiacenti. Ogni quadrato contiene un numero con dei pallini dipinti nell'insieme dei numeri naturali $0, 1, \dots, 6$. Ad esempio:

/// Example

For example, the following figure exhibits a particular sequence of $n = 4$ tiles.

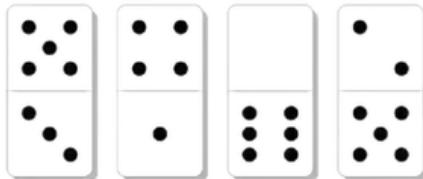


Figure 13.1: Domino pieces

Si ha quindi un ordine prefissato della tesserina (sappiamo qual è la prima, la seconda, la terza e la quarta), ma ogni tesserina ha due possibili stati: la prima può avere un valore a sinistra e uno a destra o viceversa:

Then a possible configuration could be



The weight of this configuration is $|3-1| + |4-6| + |0-5| = 9$.

Figure 13.2: Domino pieces 2

Tutte le possibili configurazioni sono 2^n in quanto: ogni tesserina ha soli due possibili stati e n perché ho n tesserine. Questo è un problema combinatorico dato che l'insieme delle soluzioni possibili è dato da tutte le possibili combinazioni degli stati delle nostre tesserine fissato il loro ordine. Chiameremo configurazione un particolare stato della nostra sequenza, ovvero uno di questi 2^n possibili stati.

Vediamo ora il concetto di **peso** per una particolare configurazione. Per peso intendiamo la **somma dei valori assoluti delle differenze tra due tesserine adiacenti**, come si può vedere nella figura precedente. Il costo della configurazione, in questo caso è pari a 9. Quello che bisogna essere in grado di fare è individuare all'interno del set delle 2^n possibili configurazioni quella che ha **peso minimo**.

ATTENZIONE: la sequenza delle nostre tesserine non può cambiare e questo rende il problema più semplice da risolvere. In particolare l'esercizio ci chiede di dimostrare che il problema sia rappresentabile o trasformabile in un problema di *cammino minimo su di un grafo diretto*. Utilizzando tale idea dobbiamo risolvere il problema in esempio, ovvero con le 4 tesserine cercare una possibile combinazione che minimizzi il loro peso.

Modelling the problem as a shortest path problem

Abbiamo visto che ogni tesserina ha due possibili stati:

1. Modelling the problem as a shortest path problem

In order to address the first point, notice that for each tile there are two possible positions. Thus, we can create a graph in which there are two nodes for each tile, associated to the possible orientations that we named position (a) and (b) respectively.

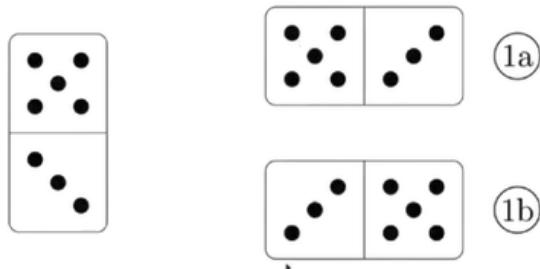


Figure 13.3: Domino pieces 3

Avremo quindi due stati:

- 1a
- 1b

In questo modo possiamo, per ciascuna di queste tesserine considerare due possibili nodi/stati di un grafo. Avremo i nodi:

Moreover, the associated cost of this choice depends only on the positions of its respective neighbors; therefore, we can link each position node with those of their neighbors labelling the arcs with the respective cost of the choices.

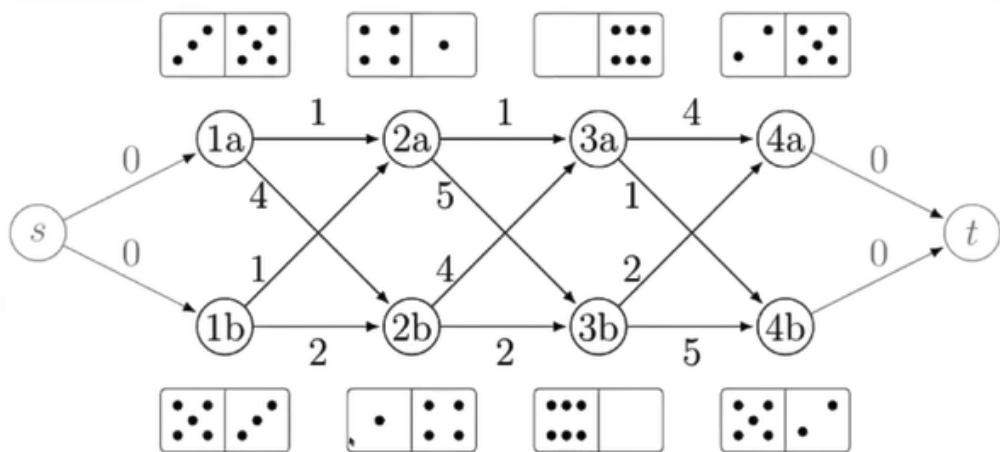


Figure 13.4: Domino pieces 4

La configurazione della tessera del domino è riportata sul grafico per rendere il mapping tra il problema iniziale e quello su grafo più intuitivo.

Gli **archi** ci dicono che effettuo una scelta: se mi trovo in 1a ho fatto una scelta per mettere la prima tesserina in tale modo; se da 1a passo a 1b significa che ho scelto di posizionare la tessera col un solo punto a sinistra. Quindi la scelta da fare, ovvero il passaggio da una possibile configurazione ad un'altra, è il valore/peso sull'arco e il valore del peso, cioè la differenza tra il valore 5 n 1a ed 1 in 2b (ovvero il valore assoluto della differenza tra due facce adiacenti). Fatto questo per ogni possibili stato delle tesserine si ottiene il grafo. Quello che abbiamo ottenuto è un grafo che partendo da uno dei possibili stati 1a o 1b posso, scegliendo un cammino su questo grafo, individuare in maniera univoca una configurazione.

Per trasformare questo problema in un problema di cammino minimo dobbiamo includere il primo set di nodi e l'ultimo set di nodi (quello relativo alla tesserina 1 e alla tesserina 4) all'interno di un processo decisionale in cui abbiamo una decisione da prendere (dove le **decisioni** sono gli **archi**). Per far questo aggiungiamo i nodi di **starting point** e di **termination point** e poi quattro archi, due uscenti da s e due entranti in t che hanno peso **nullo**: la loro identificazione non ha effetto immediato sul costo della loro configurazione. Infatti, non hanno effetto perché si dovrebbe calcolare usando il primo lato della tesserina che però non esiste. Lo stesso vale per l'ultimo nodo: abbiamo due archi ma nessun peso associato. In questo modo, noi possiamo facilmente rappresentare il problema come un problema di percorso minimo: possiamo cercare quell'insieme di archi che mi porta da s a t con un cammino di costo minimo, perché la somma dei pesi degli archi è minima. I nodi s e t sono definiti **nodi dummy**.

Come si potrebbe **definire** un percorso/path su di un grafo fra due nodi? Quali sono le sue caratteristiche?

Ogni nodo del percorso, tranne quello iniziale e quello finale, deve avere un arco entrante e uno uscente. Questo impedisce che ci siano dei cicli. Se invece di parlare di nodi parlassimo di archi dovremmo dire che dato un nodo, o ha un arco entrante e uno uscente oppure nessuna delle due cose. Scegliendo un percorso piuttosto che l'altro, nel nostro esempio, 4 nodi verranno considerati mentre i restanti 4 verranno esclusi (non considerando il nodo start e lo stop). Per i nodi che vengono esclusi devono valere le seguenti proprietà (bisogna impostare una condizione per quei nodi non presenti sul cammino): bisognerebbe impostare che ogni nodo abbia al massimo un arco entrante e un arco uscente e nulla più.

Proviamo quindi a formulare un modello matematico di ottimizzazione per questo problema:

- Cominciamo dai **nodi**;
- Poi abbiamo gli **archi** relativi alle decisioni da prendere, ovvero le possibili combinazioni;
- Infine, abbiamo una serie di pesi associati agli archi;

Formuliamo un insieme di **variabili decisionali** per questo problema: (quali decisioni dobbiamo prendere?)

- Decidere quali **archi** fanno parte del percorso e chi non ne fa parte. Questo si modella in maniera naturale con una variabile binaria. Possiamo quindi creare una variabile binaria:

$$x_{ij} = \begin{cases} 1, & \text{if arc } (i,j) \text{ is in the path} \\ 0, & \text{otherwise} \end{cases} \quad (13.1)$$

Dove (i,j) rappresenta una coppia di nodi sul nostro percorso collegati da un arco e questa variabile assumerà valore 1 se quell'arco è parte del percorso, 0 altrimenti.

- Per quanto riguarda la **funzione obiettivo** equivale alla somma dei costi associati agli archi moltiplicato per la variabile che mi dice se l'arco è attivo oppure no, ovvero:

$$\sum_{(i,j) \in A} c_{ij} x_{ij} \quad (13.2)$$

Ciò che rimane è solo il costo del cammino che abbiamo definito. E' necessario però imporre alcuni vincoli affinché il sottoinsieme di archi che sceglieremo all'interno del nostro grafo sia un cammino.

- Quindi guardiamo ai **vincoli**:

- Il numero di archi entranti deve essere uguale al numero di archi uscenti per tutti i nodi del grafo, meno che per il nodo start e stop. La differenza tra l'arco entrante e l'uscente deve essere uguale a 0:

$$\sum_{i|(i,j)} x_{ij} - \sum_{k|(j,k)} x_{jk} = 0, \quad \forall j \in V - s, t \quad (13.3)$$

Tuttavia, questo non è sufficiente a garantire l'esistenza di un arco singolo incoming e di un singolo outgoing. Potrebbero esserci situazioni in cui abbiamo due archi entranti e due uscenti. Per questo motivo dobbiamo limitare ad 1 il numero massimo di archi entranti e di archi uscenti per un certo nodo. Possiamo quindi scrivere:

$$\begin{aligned} \sum_{i|(i,j)} x_{ij} &\leq 1, \quad j \in V - s, t \\ \sum_{j|(i,j)} x_{ij} &\leq 1, \quad i \in V - s, t \end{aligned} \quad (13.4)$$

Dove si limita ad 1 l'insieme degli archi entranti in un nodo, perché lo stiamo facendo per ogni j supponendo che l'arco vada da i a j (quindi se la somma di un certo nodo per tutti gli archi entranti può essere 0 o al massimo 1). La stessa cosa in questo caso, la somma degli archi entranti e degli archi uscenti deve essere al massimo 1;

- Infine, dobbiamo considerare i nodi speciali, ovvero il nodo start e stop: per questi due casi vale la questione che il numero degli archi uscenti da s deve essere uguale ad 1 (esattamente un arco può essere scelto) e il numero totale di archi entranti in t deve essere uguale ad 1:

$$\begin{aligned} \sum_{j|(s,j) \in A} x_{sj} &= 1 \\ \sum_{i|(i,t) \in A} x_{it} &= 1 \end{aligned} \quad (13.5)$$

Mettendo insieme tutto quanto si ottiene il seguente problema di cammino minimo (perché vogliamo minimizzare la somma dei pesi sul nostro arco) su grafo:

A mathematical formulation associated with the problem is:

$$\begin{aligned}
 \min \quad & \sum_{(i,j) \in A} c_{ij} x_{ij} \\
 \text{st} \quad & \sum_{i|(i,j)} x_{ij} - \sum_{k|(j,k)} x_{jk} = 0, \quad \forall j \in V - \{s, t\}, \\
 & \sum_{i|(i,j)} x_{ij} \leq 1, \quad \forall j \in V - \{s, t\}, \\
 & \sum_{j|(i,j)} x_{ij} \leq 1, \quad \forall i \in V - \{s, t\}, \\
 & \sum_{j|(s,j) \in A} x_{sj} = 1, \\
 & \sum_{i|(i,t) \in A} x_{it} = 1 \\
 & x_{ij} \geq 0, \quad \forall (i,j) \in A.
 \end{aligned}$$

Figure 13.5: Formulazione matematica del problema

Abbiamo quindi dimostrato il primo punto dell'esercizio, ovvero dimostrare che il problema poteva essere rappresentato tramite grafo e poi modellare tale problema usando un modello di programmazione lineare intera.

A questo punto lo si risolve tramite R con il Solver lpSolveAPI.

Il problema che abbiamo modellato era un problema di programmazione lineare binaria, quindi con variabili dummy che viene risolto con l'algoritmo del Branch & Bound. Il Branch & Bound è un algoritmo **non polinomiale rispetto alla dimensione del problema**, che significa che si tratta di un algoritmo che richiede tempo per essere risolto e non scala bene per dati molto grandi (man mano che aggiungiamo un nodo/arco al grafo otteniamo una problema che è esponenzialmente più difficile da risolvere (2^n)).

Questo non vuol dire che il problema di cammino minimo può essere anche risolto con l'algoritmo di **Dijkstra's**: è un algoritmo polinomiale nel tempo (che impiga cioè un tempo polinomiale per essere risolto rispetto alla dimensione del problema). E' implementato in molte librerie per grafi: in R si può usare la libreria *igraph* (esiste l'equivalente in Python). Tale procedimento può essere visto nell'esercizio analogo.

Ex. 11 - Network modelling: Min Cost Flow Problem

Fa riferimento ad un problema classico della logistica: distribuire una certa merce attraverso una rete di comunicazione (ad esempio stradale) a costo minimo.

Si ha un'azienda che deve organizzare la sua produzione e ha una fabbrica/centro di produzione unico denotato dal numero 1. I centri di distribuzione vengono invece denotati con i numeri 2, 3, 4, 5. Ogni distribution center ha una sua Demand, ovvero una certa quantità di prodotto. La domanda relativa a questi 4 centri di distribuzione è 4, 2, 12 e 2 tonnellate di prodotto al mese rispettivamente (il centro 2 chiede 4 tonnellate al mese etc.). Quello che vogliamo fare è distribuire la produzione dando a ciascuno di questi centri quello che richiedono.

Osserviamo che il centro di produzione 1 deve essere in grado di far fronte a tutte le richieste dei centri di distribuzione. Nella tabella seguente definita **tabella origine-destinazione** (dal nodo 1 posso andare al 3 e al 4; dal nodo 2 posso andare solo al 3 e al 4; dal nodo 3 posso andare solo al nodo 5; dal nodo 4 posso andare solo al 5). Ho quindi un'indicazione di come sono collegati tra di loro i centri di

distribuzione e in più ho dei numeri. Questi numeri si leggono nel seguente modo:

- Il primo numero (all'interno della tabella, tra 1 e 2) rappresenta il costo di distribuzione per tonnellata, cioè il costo relativo ad inviare 1 tonnellata di prodotto su quel particolare arco/cammino (ovvero, se mando 1 tonnellata di prodotto 1 a 2 allora pagherò il costo unitario, ovvero 1.)
- Il secondo numero ci dice che su quella particolare rotta non ho nessun obbligo ad inviare qualcosa, se uso quella rotta (lo potremmo considerare come **vincolo di bootstrap** perché in altri casi questo vincolo è attivo, come nel caso dell'arco tra 2 e 3. Cioè se decido di spostare qualcosa da 2 a 3 deve essere almeno di due tonnellate/unità, altrimenti quella particolare rotta non è conveniente o non si può attivare);
- Riguardo il terzo numero, questa è la capacità della rotta/arco: in alcuni casi vale $+\infty$, in altri casi è limitato. Cioè posso mandare quanto prodotto voglio tra 1 e 2, ma tra 1 e 3 posso mandare al massimo 4 unità di prodotto. Questo è un vincolo superiore per quanto riguarda la capacità di una certa rotta.

The following table are reported the other data of the problem; for each pair *origin-destination* a triple is give with the following values:

- **per-ton distribution cost**, that is the cost for sending a ton of product over the routes between an origin a destination,
- **lower bound** on the total amount of product to be sent in the routes between an origin a destination (**bootstrap**), and
- **upper bound** on the total amount of product sent in the routes between an origin a destination (**route capacity**).

Origin → Destination	2	3	4	5
1	(1,0, $+\infty$)	(8,0,4)	(7,0, $+\infty$)	-
2	-	(6,2,4)	(4,0,13)	(7,0,1)
3	-	-	-	(2,0,2)
4	-	-	-	(1,0,2)

The problem requires to find the **minimum cost product flow** that satisfies

- the demands of the distribution centers,
- constraints on the routes.

Figure 13.6: Tabella origine-destinazione

Questo problema richiede di individuare il flusso di prodotto a costo minimo che dia ad ogni centro di distribuzione il prodotto pari alla sua Demand e che identifichi le rotte. Cosa dobbiamo decidere?

- Quanto muovere, ovvero quanto prodotto mandare verso ogni destinazione;
- Ma non tutte le destinazioni sono direttamente raggiungibili: per esempio dall'origina 1 alla destinazione 5 non c'è un cammino diretto, quindi per arrivarci devo per forza di cosa portare più prodotto in un centro di distribuzione intermedio e poi da lì spostare il prodotto verso il nodo finale.

Quindi si avranno:

- Vertici da 1 a 5 (che sono i vertici del nostro grafo), ovvero $V = 1, 2, 3, 4$;
- Abbiamo le Demand dei vertici identificate con b_i , con $i \in V$;

- L'insieme degli archi, ovvero:

$$A = (i, j) \in V^2 = (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (2, 5), (3, 5), (4, 5) \quad (13.6)$$

ovvero l'insieme degli archi orientati nel grafo, dove i è un vertice e j è un'altro vertice ma non tutti i punti del prodotto cartesiano $V \times V$ collegati, ma soltanto un insieme di questi;

- Per ogni arco abbiamo un **costo di trasporto** che chiamiamo c_{ij} ;
- Si hanno poi due valori che sono il **lower bound** e l'**upper bound** riguardo la quantità trasportata su quell'arco;
- Infine, la **variabile decisionale** è la quantità di prodotto che viene trasferito attraverso un arco.

Il grafo può essere così visualizzato:

The problem can be represented graphically as:

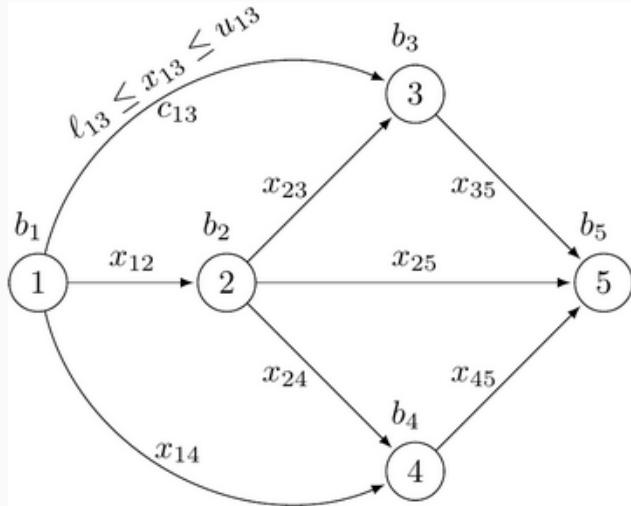


Figure 13.7: Grafo 1

Gli archi sono denotati con il numero da 1 a 5, mentre gli archi sono denotati in tre modi:

- Hanno un'etichetta relativa al costo dell'arco;
- Un'etichetta per il **lower bound**;
- Un'etichetta per l'**upper bound**;
- Si ha poi la **variabile associata all'arco** definita x_{ij} .

Una volta che abbiamo la rappresentazione formuliamo il nostro problema.

Le **variabili decisionali** sono le x_{ij} che sono quelle sugli archi.

La **funzione obiettivo** sarà uguale a? Si tratta innanzitutto di un problema di **minimizzazione**: ai nostri depositi vogliamo dare **esattamente** ciò che ci hanno richiesto. E vogliamo farlo minimizzando il costo di trasporto. Come? Considerando i **costi sugli archi moltiplicati per la quantità che stiamo trasferendo**, in quanto x_{ij} rappresenta la quantità di prodotto che stiamo trasferendo. Questa quantità di prodotto è un numero intero o frazionario? Bisogna sempre chiedersi che tipo di prodotto stiamo trasportando: se non abbiamo indicazioni specifiche sulla non divisibilità del prodotto, o dalla limitata divisibilità del prodotto, allora il problema è un **problema continuo** (non c'è motivo di renderlo più

difficile e rappresentarlo intero).

NOTA: quando faremo l'esame o si dovranno risolvere problemi nella vita reale, l'importante è chiedersi se nel problema c'è qualche suggerimento sul fatto che il problema sia intero o continuo. Quindi cercare di definire il tipo della variabile come uno dei primi passi della modellazione.

Guardiamo ai **vincoli**:

- Ogni centro di distribuzione deve vedere soddisfatta la propria Demand. Un centro di distribuzione può essere parte di un cammino. Immaginiamo che la Demand del centro 5 venga soddisfatta passando per il centro 4: ciò che è entrante in 4 (ovvero, x_{14}) più x_{24} , cioè tutto ciò che entra, meno ciò che esce e che è diretto a 5, ovvero x_{45} deve essere uguale alla Demand di questo centro. Questo lo si vede dal seguente vincolo:

$$\sum_{i:(i,j) \in A} x_{ij} - \sum_{k:(j,k) \in A} x_{jk} = b_j, \quad \forall j \in V \quad (13.7)$$

Ovvero la somma della quantità di prodotto entrante in un nodo, meno la somma di un prodotto uscente da un nodo deve essere esattamente uguale alla Demand di quel nodo;

- Si hanno poi vincoli sulla capacità dei nostri archi: abbiamo che le variabili decisionali che decidono quanto prodotto trasportare su di un certo arco, devono essere limitate inferiormente dal nostro **lower bound** e limitate superiormente dal nostro **upper bound**, ovvero:

$$l_{ij} \leq x_{ij} \leq u_{ij}, \quad \forall (i, j) \in A \quad (13.8)$$

Il nostro modello finale sarà pari a:

The resulting **mathematical formulation** (LP) associated with the problem:

$$\begin{aligned} \min \quad & \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \text{st} \quad & \sum_{i:(i,j) \in A} x_{ij} - \sum_{k:(j,k) \in A} x_{jk} = b_j, \quad \forall j \neq 1 \in V, \\ & l_{ij} \leq x_{ij} \leq u_{ij}, \quad \forall (i, j) \in A, \\ & x_{ij} \geq 0, \quad \forall (i, j) \in A. \end{aligned}$$

Figure 13.8: Mathematical Formulation

That is:

$$\begin{array}{ll}
 \text{minimize} & x_{12} + x_{13} + 7x_{14} + 6x_{23} + 4x_{24} + 7x_{25} + 2x_{35} + x_{45} \\
 \text{subject to} & \\
 \begin{array}{rccccccl}
 x_{12} & & -x_{23} & -x_{24} & -x_{25} & & = 4 \\
 x_{13} & & +x_{23} & & & -x_{35} & = 2 \\
 x_{14} & & & +x_{24} & & & -x_{45} = 12 \\
 & & & & x_{25} & +x_{35} & +x_{45} & = 2
 \end{array} \\
 \hline
 \begin{array}{lll}
 0 \leq & x_{12} & \leq \text{Inf} \\
 0 \leq & x_{13} & \leq 4 \\
 0 \leq & x_{14} & \leq \text{Inf} \\
 2 \leq & x_{23} & \leq 4 \\
 0 \leq & x_{24} & \leq 13 \\
 0 \leq & x_{25} & \leq 1 \\
 0 \leq & x_{35} & \leq 2 \\
 0 \leq & x_{45} & \leq 2
 \end{array}
 \end{array}$$

we now rename the variables and put the model in matricial form:

$$\text{minimize } (1 \ 8 \ 7 \ 6 \ 4 \ 7 \ 2 \ 1)(y_1 \ y_2 \ y_3 \ y_4 \ y_5 \ y_6 \ y_7 \ y_8)^t$$

$$\text{s.t. } \left[\begin{array}{cccccccc} 1 & 0 & 0 & -1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array} \right] \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 12 \\ 2 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \leq \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{pmatrix} \leq \begin{pmatrix} 8 \\ 1 \\ 7 \\ 6 \\ 4 \\ 7 \\ 2 \\ 1 \end{pmatrix}$$

Notice that this problem can be expressed with a (continuous) linear programming model.

Figure 13.9: Mathematical Formulation 2

Successivamente lo si risolve in R.

Chapter 14

LEZIONE 12 - Webex Lecture n. 9 (26/04/2021) - Non linear Programming Problems

Continuiamo da dove ci eravamo lasciati nella lezione precedente all'esercitazione appena conclusa, ovvero:

5. Generalized Network Flow Problems:

Nel caso in cui ci siano delle perdite o guadagni di flusso lungo gli archi dobbiamo modellare il problema andando ad aggiungere un fattore moltiplicativo che rappresenti questa cosa. Nel trasporto da un nodo ad un altro si ha un guadagno o una perdita di flusso.

Coal Bank Hollow Recycling

Processo di riciclaggio di carta:

Coal Bank Hollow Recycling

Material	Process 1		Process 2		Supply
	Cost	Yield	Cost	Yield	
Newspaper	\$13	90%	\$12	85%	70 tons
Mixed Paper	\$11	80%	\$13	85%	50 tons
White Office Paper	\$9	95%	\$10	90%	30 tons
Cardboard	\$13	75%	\$14	85%	40 tons

Pulp Source	Newsprint		Packaging Paper		Print Stock	
	Cost	Yield	Cost	Yield	Cost	Yield
Recycling Process 1	\$5	95%	\$6	90%	\$8	90%
Recycling Process 2	\$6	90%	\$8	95%	\$7	95%
Demand	60 tons		40 tons		50 tons	

Figure 14.1: Coal Bank Hollow Recycling

Si hanno due processi diversi che hanno costi e rendimenti diversi per ciascuno dei materiali grezzi e per ciascun processo. L'ultima colonna riporta la disponibilità dei materiali. Questi materiali sono processati e questa polpa viene utilizzata per produrre altri materiali. Anche in questo caso, si hanno diversi costi e rendimenti per ogni nuovo materiale prodotto dalla polpa.

Qui è più difficile capire se la disponibilità è sufficiente a soddisfare la domanda (non possiamo fare solo la somma, dobbiamo capire qual è l'effettivo flusso che attraversa la rete). Si risolve il problema e

si vede se la soluzione è ammissibile.

Network for Recycling Problem

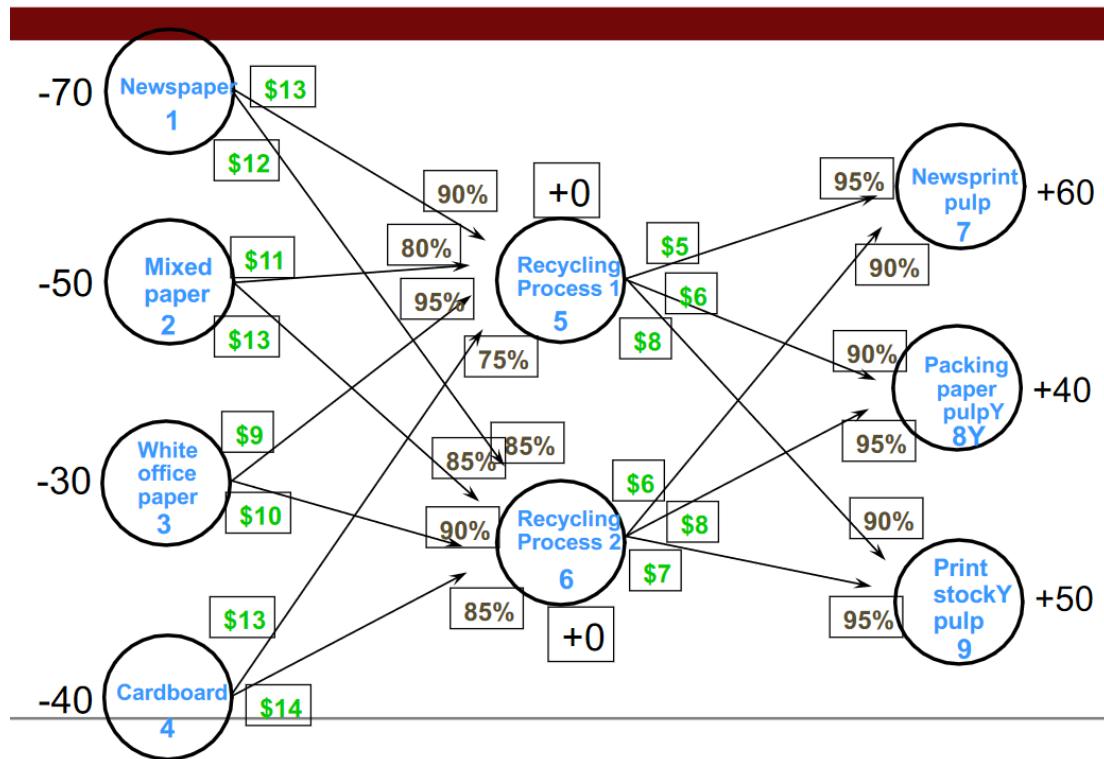


Figure 14.2: Grafo

A sinistra si una offerta/disponibilità; il nodo V e il nodo VI rappresentano il processo di riciclaggio e poi si hanno i tre nodi che rappresentano i tre nodi finali. Sugli archi si vedono sia il costo che il rendimento del processo.

La funzione obiettivo sarà di minimizzare i costi, sia del processo da materiali grezzi a polpa di carta sia da polpa di carta a materiali finiti:

Defining the Objective Function

Minimize total cost.

$$\text{MIN: } 13X_{15} + 12X_{16} + 11X_{25} + 13X_{26} + 9X_{35} + 10X_{36} + 13X_{45} + \\ + 14X_{46} + 5X_{57} + 6X_{58} + 8X_{59} + 6X_{67} + 8X_{68} + 7X_{69}$$

Figure 14.3: Objective Function

Si hanno quindi tante variabili quante sono gli archi e ogni variabile è moltiplicata per il costo associato all'arco.

Si hanno poi per i nodi che rappresentano il materiale grezzo che quello che esce (meno l'outflow deve essere maggiore/uguale della capacità):

Defining the Constraints-I

- Raw Materials

$$-X_{15} - X_{16} \geq -70 \quad \} \text{ node 1}$$

$$-X_{25} - X_{26} \geq -50 \quad \} \text{ node 2}$$

$$-X_{35} - X_{36} \geq -30 \quad \} \text{ node 3}$$

$$-X_{45} - X_{46} \geq -40 \quad \} \text{ node 4}$$

Figure 14.4: Defining Constraint

Dopodiché abbiamo i nodi V e il nodo VI:

Defining the Constraints-II

- Recycling Processes

$$+0.9X_{15} + 0.8X_{25} + 0.95X_{35} + 0.75X_{45} - X_{57} - X_{58} - X_{59} \geq 0 \quad \} \text{ node 5}$$

$$+0.85X_{16} + 0.85X_{26} + 0.9X_{36} + 0.85X_{46} - X_{67} - X_{68} - X_{69} \geq 0 \quad \} \text{ node 6}$$

Figure 14.5: Defining Constraints 2

Questi sono **nodi di transshipment** dove si hanno archi in ingresso e archi in uscita e non c'è una richiesta o disponibilità su questi nodi, sono quindi di passaggio. Analogamente ai problemi di trasporto abbiamo l'inflow meno l'outflow maggiore/uguale a 0. L'inflow deve però essere moltiplicato per l'arco e quindi al nodo V non arriva più tutto quello che esce dal nodo I, ma arriva quello che esce dal nodo I diminuito del 10% perché il rendimento è il 90%.

Si hanno poi i nodi VII, VIII, IX:

Defining the Constraints-III

- Paper Pulp

$$\begin{aligned}
 +0.95X_{57} + 0.90X_{67} &\geq 60 \quad \} \text{ node 7} \\
 +0.90X_{57} + 0.95X_{67} &\geq 40 \quad \} \text{ node 8} \\
 +0.90X_{57} + 0.95X_{67} &\geq 50 \quad \} \text{ node 9}
 \end{aligned}$$

Figure 14.6: Defining Constraints 3

Quello che entra in questi nodi viene moltiplicato per il relativo rendimento deve soddisfare la richiesta e quindi deve essere maggiore/uguale al valore rhs.

La soluzione finale sarà data da:

Problem Solution

Flow From Node		Yield		Flow Into Node		Cost
43.4	1 Newspaper	0.90		39.1	5 Process 1	\$13
26.6	1 Newspaper	0.85		22.6	6 Process 2	\$12
50.0	2 Mixed Paper	0.80		40.0	5 Process 1	\$11
0.0	2 Mixed Paper	0.85		0.0	6 Process 2	\$13
30.0	3 White Office	0.95		28.5	5 Process 1	\$9
0.0	3 White Office	0.90		0.0	6 Process 2	\$10
0.0	4 Cardboard	0.75		0.0	5 Process 1	\$13
35.4	4 Cardboard	0.85		30.1	6 Process 2	\$14
63.2	5 Process 1	0.95		60.0	7 Newsprint	\$5
44.4	5 Process 1	0.90		40.0	8 Packaging	\$6
0.0	5 Process 1	0.90		0.0	9 Print Stock	\$8
0.0	6 Process 2	0.90		0.0	7 Newsprint	\$6
0.0	6 Process 2	0.95		0.0	8 Packaging	\$8
52.6	6 Process 2	0.95		50.0	9 Print Stock	\$7
					Total Cost	\$3,149

Figure 14.7: Solution

In questo caso è difficile sapere a priori se il problema è ammissibile o no, ovvero se la disponibilità è sufficiente per soddisfare la domanda perché non basta fare la somma delle disponibilità e verificare che sia maggiore/uguale alla richiesta. Ma bisogna risolvere il problema perché il calcolo è più complicato e dipende da come viene diretto il flusso nell'arco.

6. The Maximum Flow Problem:

Problema del massimo flusso su una rete. In questo caso gli archi hanno una capacità **minima** e una **massima** (hanno dei limiti inferiori e superiori) e quello che voglio determinare è quanto flusso può attraversare la rete di distribuzione/traffico. In questo caso non esplicito delle disponibilità all'origine o delle richieste alla destinazione, ma semplicemente voglio capire qual è il flusso massimo che può attraversare la rete. (es. problemi di distribuzione di reti idriche, petrolio, gas etc.).

Nel seguente esempio abbiamo a che fare con un pozzo petrolifero:

The Northwest Petroleum Company

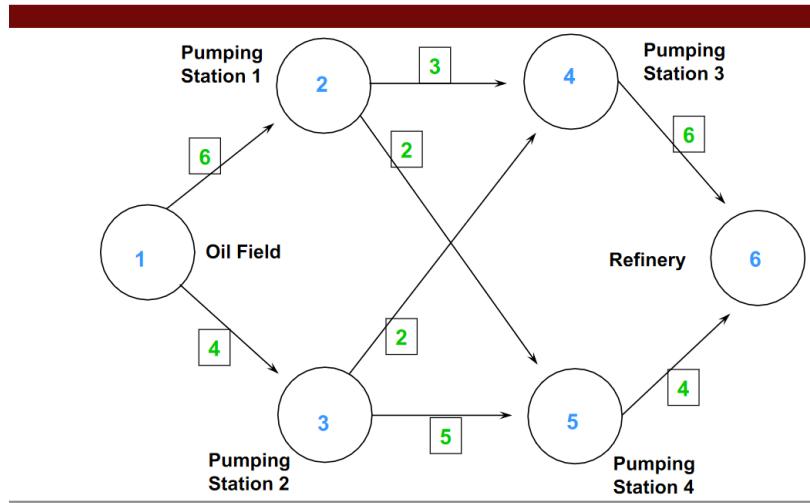


Figure 14.8: The Northwest Petroleum Company

Il pozzo petrolifero viene rappresentato dal nodo I che manda il petrolio nelle stazioni II e III e dalla stazione II e III il petrolio può essere mandato alle stazioni IV e V per poi essere mandato alla raffineria al nodo VI: c'è una rete di distribuzione con delle stazioni di pompaggio che fanno in modo che il petrolio arrivi dal campèo petrolifero fino alla raffineria passando dalle stazioni di pompaggio. La **capacità massima degli archi** è il numero in verde vicino all'arco e quello che voglio capire è qual è il flusso massimo che può arrivare dal nodo I al nodo VI. In questo caso non ho più una quantità che voglio spedire dal nodo I al nodo VI ma voglio capire qual è la quantità massima. La differenza è che questa è una **variabile di decisioni** e non più un dato disponibile del problema. tale variabile di decisione viene rappresentata mettendo un arco fittizio dal nodo I al nodo VI (nei problemi di flusso fin'ora le variabili decisionali si trovavano sempre sugli archi): aggiungo un altro arco quindi dal nodo I al nodo VI con costo di attraversamento 0 e la variabile X_{61} diventa una variabile aggiuntiva del problema ed è quella che voglio massimizzare: voglio massimizzare il flusso che attraversa l'arco VI-I.

The Northwest Petroleum Company

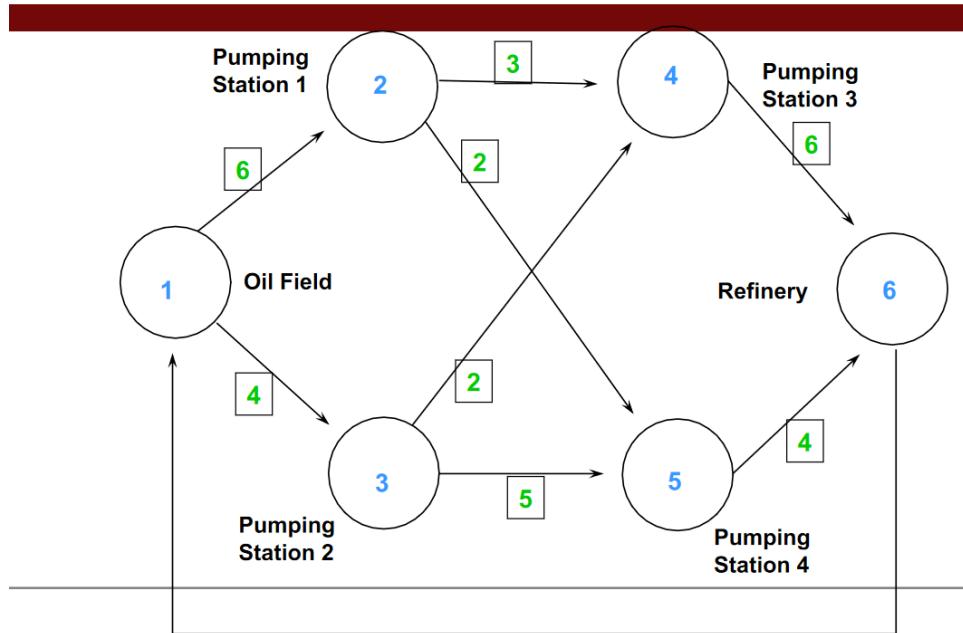


Figure 14.9: The Northwest Petroleum Company nodes VI-I

Per determinare il flusso massimo uso quindi questo trucchetto. Quindi il flusso che attraversa la rete è rappresentato dal nodo VI-I che voglio massimizzare e poi ho gli altri vincoli analoghi a quanto visto prima: la differenza tra tutto quello che entra nel nodo I e quello che esce deve essere uguale a 0 (e tutti gli altri vincoli di bilancio). Ho quindi un vincolo per ogni nodo (vincoli di bilanciamento del flusso). Oltre a questi abbiamo degli upper bound sulle variabili di decisione perché abbiamo un flusso limitato che può attraversare gli archi:

$$\text{MAX: } X_{61}$$

$$\begin{aligned} \text{Subject to: } & +X_{61} - X_{12} - X_{13} = 0 \\ & +X_{12} - X_{24} - X_{25} = 0 \\ & +X_{13} - X_{34} - X_{35} = 0 \\ & +X_{24} + X_{34} - X_{46} = 0 \\ & +X_{25} + X_{35} - X_{56} = 0 \\ & +X_{46} + X_{56} - X_{61} = 0 \end{aligned}$$

with the following bounds on the decision variables:

$$\begin{array}{lll} 0 \leq X_{12} \leq 6 & 0 \leq X_{25} \leq 2 & 0 \leq X_{46} \leq 6 \\ 0 \leq X_{13} \leq 4 & 0 \leq X_{34} \leq 2 & 0 \leq X_{56} \leq 4 \\ 0 \leq X_{24} \leq 3 & 0 \leq X_{35} \leq 5 & 0 \leq X_{61} \leq \infty \end{array}$$

Figure 14.10: Vincoli di bilancio

Ovviamente non avremo un **upper bound** per la variabile **X61** perché è quella che vogliamo massimizzare.

Optimal Solution

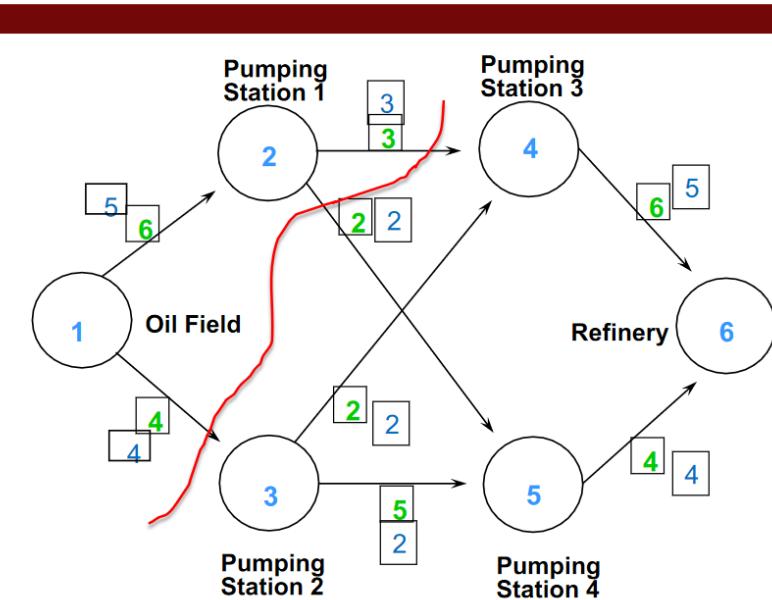


Figure 14.11: Soluzione

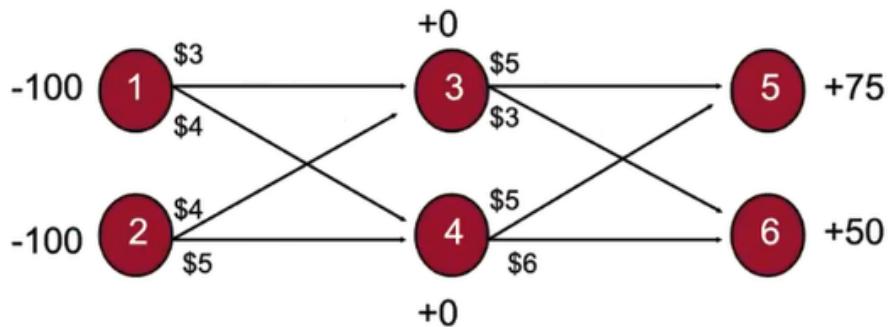
Dove in blu troviamo rappresentata la soluzione: quindi il massimo flusso che può attraversare la rete sarà uguale a tutto il flusso che raggiunge il nodo VI e sarà uguale a 9 e ovviamente il flusso che esce dal nodo I dovrà essere uguale a 9. Avremo che il nodo I-II sarà attraversato da un flusso pari a 5 e così via.

La linea rossa rappresenta un **taglio**: un taglio in un grafo divide i nodi in due parti, in due insiemi. In questo caso i due insiemi sono il nodo I e II e dall'altro insieme i nodi III, IV, V, VI. Ogni taglio in un grafo ha associata una capacità che è il flusso massimo che può attraversare questo taglio. Un'importante risultato anche teorico (che non dimostriamo) è che il massimo flusso che può attraversare la rete è equivalente alla capacità del **taglio minimo** (ovvero il taglio con capacità minima). Quindi posso tagliare la rete in vari modi; il flusso che attraversa questo taglio è dato ovviamente dalla somma delle capacità degli archi (ad esempio, in questo caso il flusso che attraversa questo taglio sarà pari a 3, più 2, più 4 = 9). Mentre se dovessimo considerare il taglio che mette in un insieme i nodi 1, 2, 3, 4 e nell'altro insieme i nodi 5, 6 avremmo una capacità pari a 13. Noi abbiamo detto che il flusso massimo che può arrivare al nodo VI è 9; se noi facciamo tutti i possibili modi in cui possiamo dividere i nodi in due sottoinsiemi e consideriamo le capacità di questi tagli, vedremo che il massimo flusso che può attraversare la rete è uguale alla capacità del taglio minimo. Questo è il classico *collo di bottiglia*: il flusso massimo che può attraversare la rete è dato dal collo di bottiglia del grafo. I due problemi sono equivalenti: se vengono formulati in modo appropriato, l'uno è il duale dell'altro; il problema del taglio minimo è il duale del problema del flusso massimo (infatti tale problema si chiama *Max Flow Min Cut - Massimo flusso minimo taglio*).

Special Modeling Considerations: Flow Aggregation

Supponiamo di avere il seguente grafo:

Special Modeling Considerations: Flow Aggregation



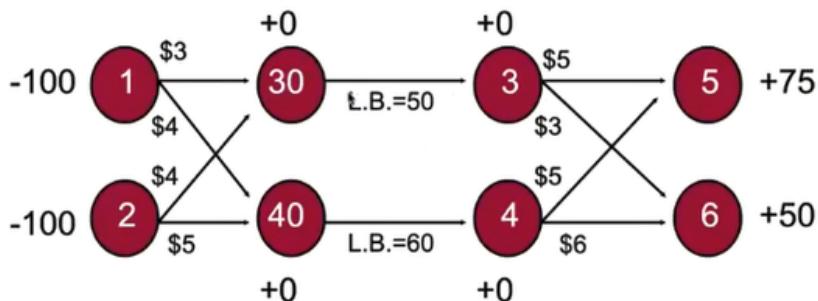
Suppose the total flow into nodes 3 & 4 must be at least 50 and 60, respectively. How would you model this?

Figure 14.12: Special Modeling Considerations

Dove nel nodo I e II abbiamo una capacità di 100 e nel nodo V e VI abbiamo una richiesta di 75 e di 50. Ma supponiamo che i nodi III e IV non siano semplicemente di passaggio ma che il flusso totale in tali nodi debba essere di 50 e 60 rispettivamente. Partendo dal fatto che in un problema come il precedente di massimo flusso su un grafo, è facile introdurre dei vincoli sugli archi tramite un lower o un upper bound, dal punto di vista modellistico. Mentre è facile mettere un lower bound o un upper bound sulla capacità di un arco, sul **nodo è più complicato**: come posso rappresentare che il flusso che arriva al nodo III e IV debba essere almeno 50 e 60 rispettivamente in modo che il modello mantenga la struttura dei vincoli che abbiamo visto? (che sono quelli dell'equazioni di bilancio che rendono il problema più semplice da risolvere)

Creiamo quindi dei nodi fittizi che si collegano al nodo III e IV dove mettiamo il lower bound sull'arco. Rappresentiamo quindi il problema come un problema di trasporto con un lower bound sulla variabile che rappresenta il flusso che va dal nodo 30 al nodo 3:

Special Modeling Considerations: Flow Aggregation

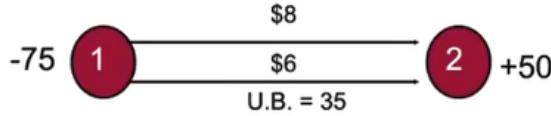


Nodes 30 & 40 aggregate the total flow into nodes 3 & 4, respectively.

Figure 14.13: Nodi aggiuntivi

Special Modeling Considerations: Multiple Arcs Between Nodes

Special Modeling Considerations: Multiple Arcs Between Nodes



Two two (or more) arcs cannot share the same beginning and ending nodes. Instead, try...

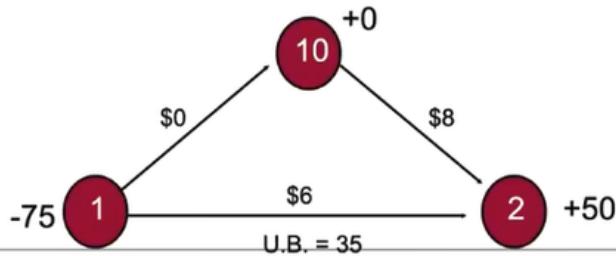
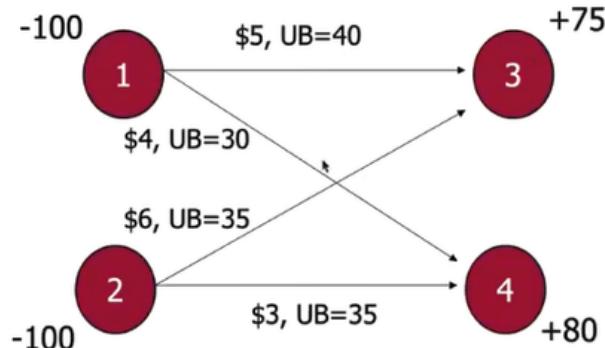


Figure 14.14: Multiple Arcs Between Nodes

Altro caso particolare: supponiamo che il flusso che viene inviato dal nodo I al nodo II abbia due costi diversi. Fino ad un massimo di 35 il costo è pari a 6, poi i costi salgono ad 8. Ovviamente non posso rappresentare questa situazione con lo stesso arco perché ogni arco deve avere associato solo un costo. Posso quindi aggiungere un **nodo fittizio** avendo quindi un arco che va dal nodo I al nodo II che costa 6 e ha un upper bound di 35 che è la prima parte del flusso, con un costo di 6. E poi il flusso aggiuntivo che verrà scelto con una priorità più bassa perché il flusso da I a II ha un costo pari a 6 e quindi cercherò prima di saturare questo flusso e poi se devo inviare ulteriore flusso dal nodo I al nodo II lo farà passare dal nodo X e questo avrà un costo unitario di 8. In questo caso riesco a rappresentare questa situazione in cui il flusso che attraversa l'arco che va da 1 a 2 avrà due costi differenziati.

Special Modeling Considerations: Capacity Restrictions on Total Supply

Special Modeling Considerations: Capacity Restrictions on Total Supply

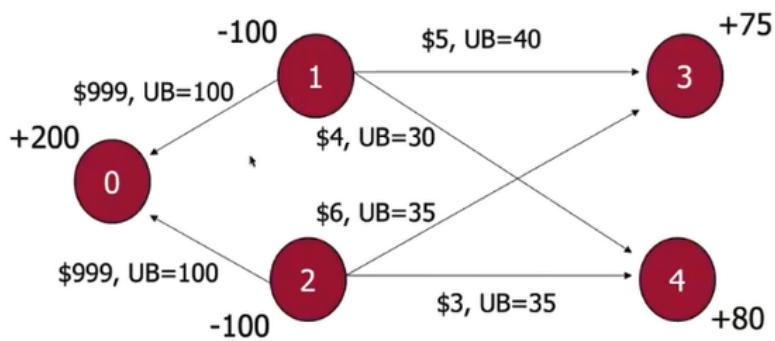


Supply exceeds demand, but the upper bounds prevent the demand from being met.

Figure 14.15: Capacity Restrictions on Total Supply

Altro caso particolare è quello in cui la disponibilità sia superiore alla domanda: in questo caso abbiamo disponibilità di 200 e una domanda di 155, ma ho dei limiti sugli archi (degli **upper bound**) che non mi permettono di soddisfare tutta la domanda di 3 e 4. In questo caso se mettessi i vincoli come abbiamo sempre messo (inflow - outflow \geq capacity/demand) e poi mettessimo gli upper bound sugli archi otterremmo un problema non ammissibile. Per trasformarlo in modo tale che sia ammissibile si aggiunge un nodo fittizio (la sorgente) che ha una capacità di 200 e vi assegno un costo molto alto dal nodo 0 al nodo I (999) e sostanzialmente verrà spedito dal nodo 0 al nodo III solo il necessario e non necessariamente tutta la quantità che è pari a 100.

Special Modeling Considerations: Capacity Restrictions on Total Supply



Now demand exceeds supply. As much “real” demand as possible will be met in the least costly way.

Figure 14.16: Capacity Restrictions on Total Supply 2

7. The Minimal Spanning Tree Problem:

L'ultimo problema su grafo che vediamo è il seguente: si tratta di una rete con n nodi; devo trovare $n - 1$ archi che connettono tutti i nodi della rete senza cicli e tutti i nodi devono essere connessi. Tale problema è costituito dal selezionare gli archi del grafo che connettono tutti i nodi senza formare cicli al minimo costo.

Supponiamo di avere un grafo di questo tipo:

Minimal Spanning Tree Example: Windstar Aerospace Company

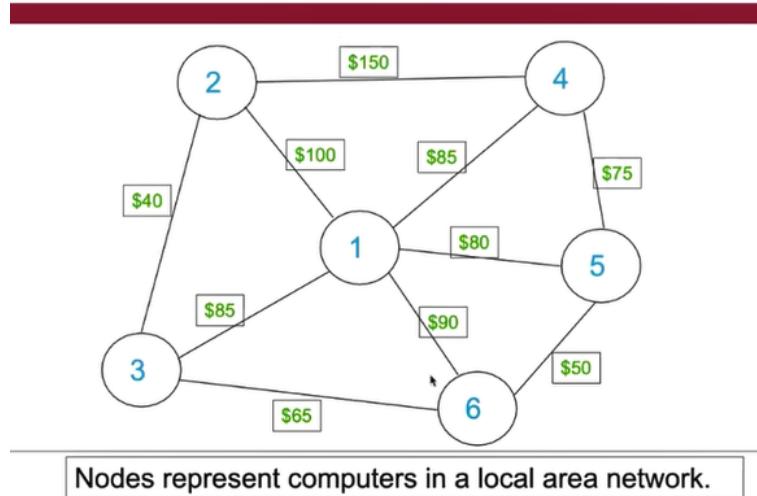


Figure 14.17: Minimal Spanning Tree

Abbiamo un grafo con 6 nodi e i possibili archi aggiunti. Supponiamo che questi rappresentino dei computer collegati in rete e vogliamo selezionare quali collegamenti attivare in modo tale che tutti i computer siano connessi minimizzando i costi di connessione. In questo caso si ha un'**euristica greedy** (greedy (ingordo)): ad ogni step considera la soluzione che è ottima localmente e questo porta alla soluzione ottima globale. Ingordo nel senso che sceglie sempre la cosa migliore e arriva comunque alla soluzione globale) per risolvere il problema:

1. Si seleziona un nodo qualsiasi della rete. Considero tutti gli archi che escono da questo nodo e seleziono quello meno costoso.
2. Aggiungo questo nodo ai nodi selezionati; considero tutti gli archi che escono dai nodi selezionati ai nodi che non sono ancora stati selezionati; scelgo quello a minimo costo e lo aggiungo;
3. Se tutti i nodi si trovano all'interno di questa sottolista mi fermo: questa è la soluzione ottimale. Altrimenti ritorno al secondo step.

Quindi, graficamente:

Solving the Example Problem - 1

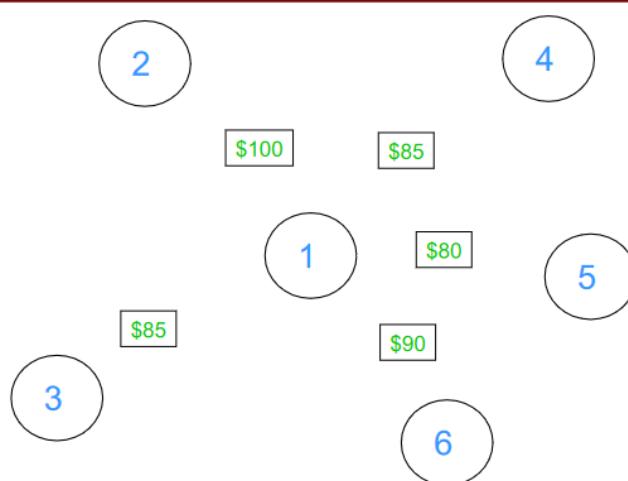


Figure 14.18: Minimal Spanning Tree 1

Solving the Example Problem - 2

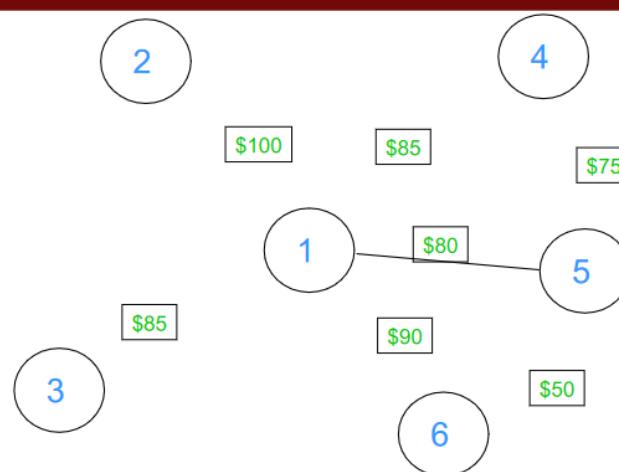


Figure 14.19: Minimal Spanning Tree 2

Solving the Example Problem - 2

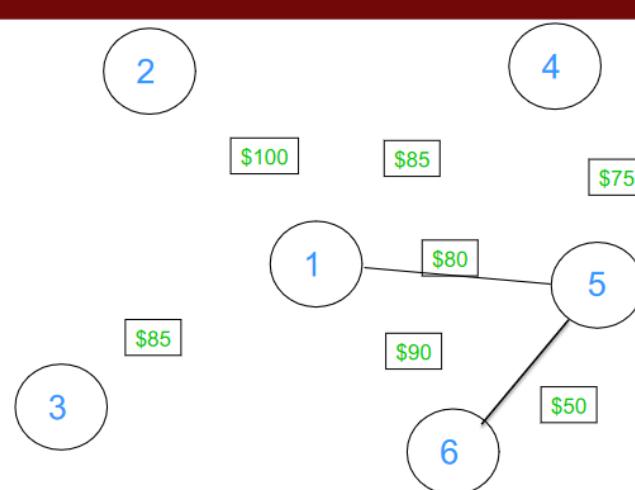


Figure 14.20: Minimal Spanning Tree 3

Solving the Example Problem - 2

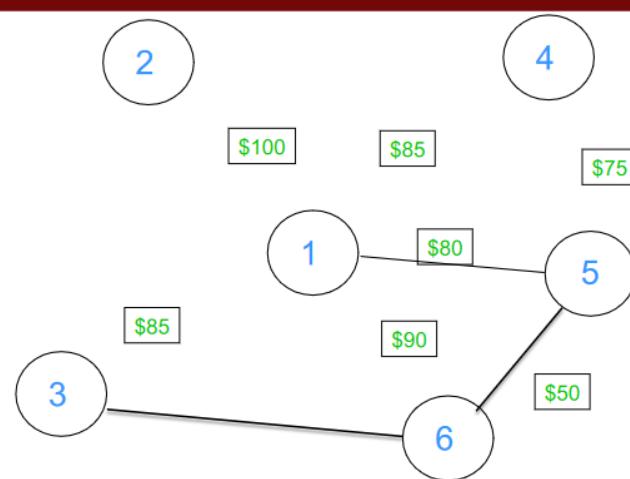


Figure 14.21: Minimal Spanning Tree 4

Solving the Example Problem - 2

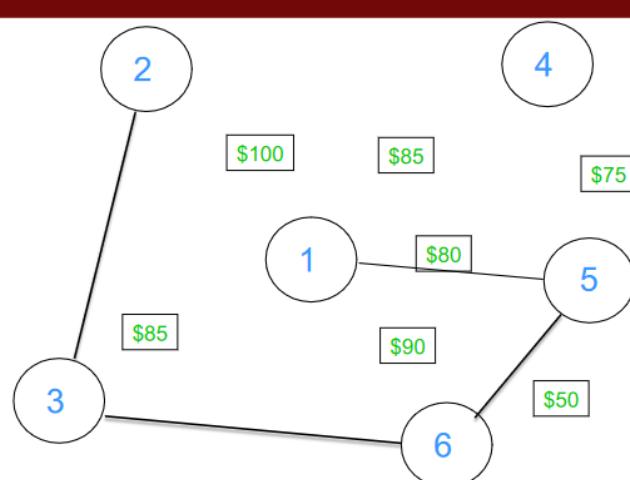


Figure 14.22: Minimal Spanning Tree 5

Solving the Example Problem - 2

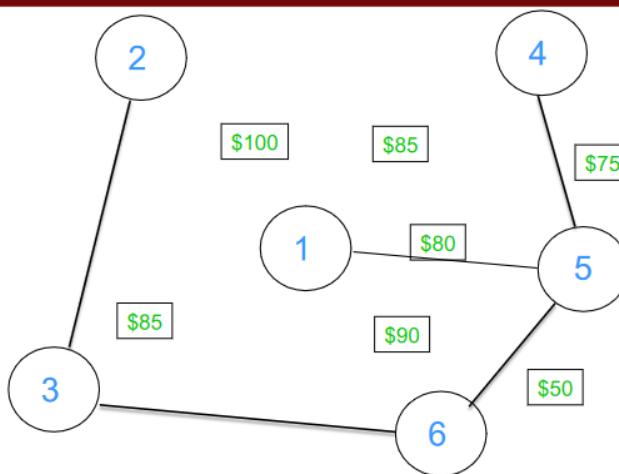


Figure 14.23: Minimal Spanning Tree 6

Ai 6 nodi ho quindi aggiunto 5 archi ($n - 1$ archi) ed esiste un cammino (gli archi non sono direzionali: li posso percorrere in entrambe le direzioni) tale che da tutti i nodi posso raggiungere tutti gli altri (tutti i nodi sono connessi e questa è la connessione a costo minimo).

Il vantaggio dei problemi su grafi è che la struttura particolare di questi problemi li rende facili da risolvere.

14.0.1 Non Linear Programming: Introduction

Introduciamo la programmazione **non lineare**. Fin'ora, nella programmazione lineare, i vincoli e la funzione obiettivo erano lineari.

In molti problemi pratici abbiamo la necessità di modellare delle relazioni che però non sono lineari e quindi ci troviamo a dover risolvere problemi di ottimizzazione che hanno la funzione obiettivo e/o i vincoli **non lineari**. Uno di questi problemi è, ad esempio, l'apprendimento dei pesi su di una rete neurale (in generale, i problemi di apprendimento in Machine Learning che devono minimizzare l'errore tra una quantità predetta e i valori osservati). Dovendo minimizzare l'errore di predizione (l'errore in genere si rappresenta con una funzione quadratica ma non certo con una funzione lineare) significa minimizzare una funzione non lineare.

Qual è il problema con la programmazione non lineare?

- I problemi NLP hanno una funzione obiettivo **non lineare** e/o uno più vincoli non lineari;
- Riguardo la loro formulazione e la loro implementazione è simile ai problemi lineari, l'unica differenza è appunto che abbiamo delle funzioni non lineari;
- La matematica che si utilizza è piuttosto differente rispetto ai problemi lineari;
- I Solver tendono a nascondere tali differenze ma è importante conoscere le difficoltà che si incontrano nel risolvere problemi di programmazione non lineare.

Se guardiamo ai seguenti esempi:

*Possible Optimal Solutions to NLPs
(not occurring at corner points)*

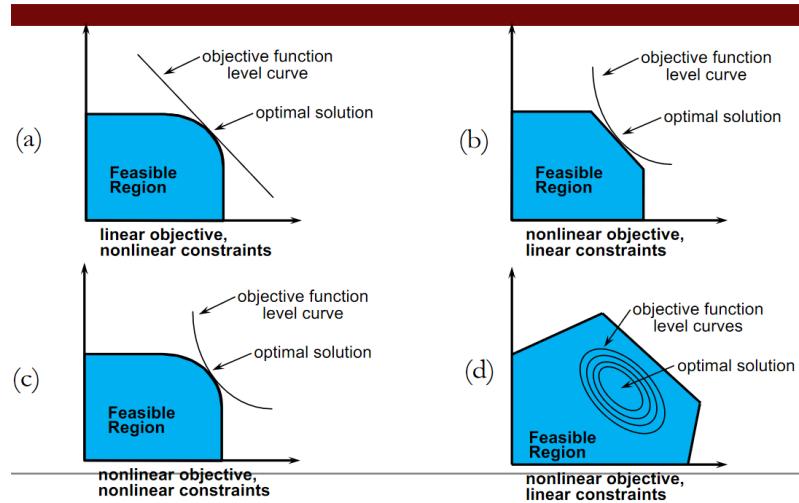


Figure 14.24: NLP

Abbiamo:

- a - In questo caso abbiamo una funzione obiettivo che è lineare e una regione ammissibile che ha un vincolo non lineare. Osserviamo che la soluzione si trova sul bordo della regione ammissibile ma non è su un vertice perché non c'è un vertice in quel punto. Quindi, l'algoritmo del simplex che basava la sua efficienza nel passare da un vertice ad un altro adiacente, in questo caso non viene utilizzato;
- b - Abbiamo una regione ammissibile costituita da vincoli lineari e una funzione obiettivo che è non lineare. Anche in questo caso vediamo che la soluzione ottima è sul bordo della regione, ma non è su un vertice;
- c - Abbiamo sia la funzione obiettivo non lineare che la regione ammissibile non lineare e ancora una volta il punto di ottimo si trova sul bordo ma non è su un vertice (il metodo del simplex non può essere utilizzato);
- d - Le curve di livello (dove, su ogni curva, il valore della funzione obiettivo è costante) di una funzione obiettivo non lineare (in questo caso quadratica) diminuiscono mano a mano che diminuisce il valore della funzione obiettivo fino ad arrivare (se stiamo minimizzando, al centro delle curve di livello). In questo caso abbiamo un problema di NLP che ha una soluzione ottima all'interno della regione ammissibile, cosa che non avevamo nella programmazione lineare.

Quindi nei problemi di NLP **non è più vero** che le soluzioni si trovino sui vertici della regione ammissibile.

Dobbiamo trovare, partendo da un punto iniziale che supponiamo sia A, un algoritmo che partendo da A trovi una direzione di miglioramento della funzione obiettivo (supponiamo che la direzione sia quella rossa).

An NLP Solution Strategy

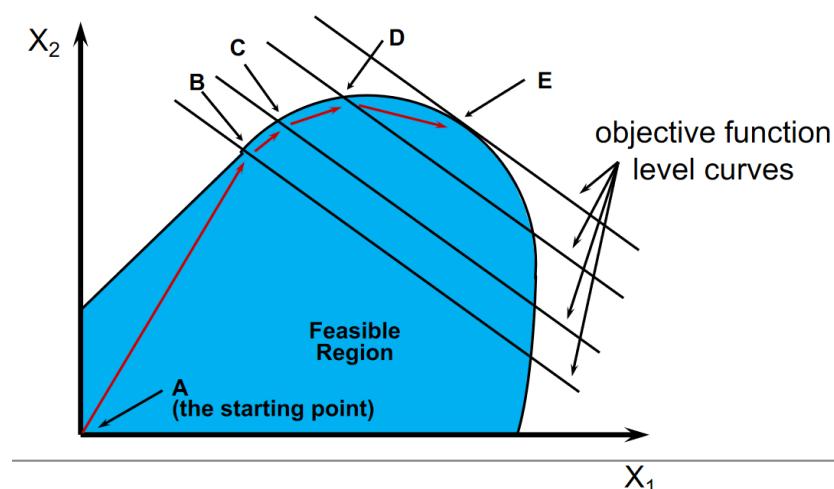


Figure 14.25: NLP solution strategy

Supponiamo per semplicità che la funzione obiettivo a mia disposizione sia lineare, quindi le rette sono le curve di livello della mia funzione obiettivo: se parto da un punto A, posso determinare una funzione di crescita della mia funzione obiettivo, mi incammino lungo quella direzione fino ad incontrare il punto B, dopodiché individuo un'altra direzione arrivando al punto C e così via fino al punto E. L'idea potrebbe essere quindi quella di trovare questa direzione e capire qual è la lunghezza del passo che posso compiere lungo questa direzione, per non uscire dalla regione ammissibile.

Qual è il problema? Che la mia regione ammissibile potrebbe **non essere convessa**, come la seguente:

Local vs. Global Optimal Solutions

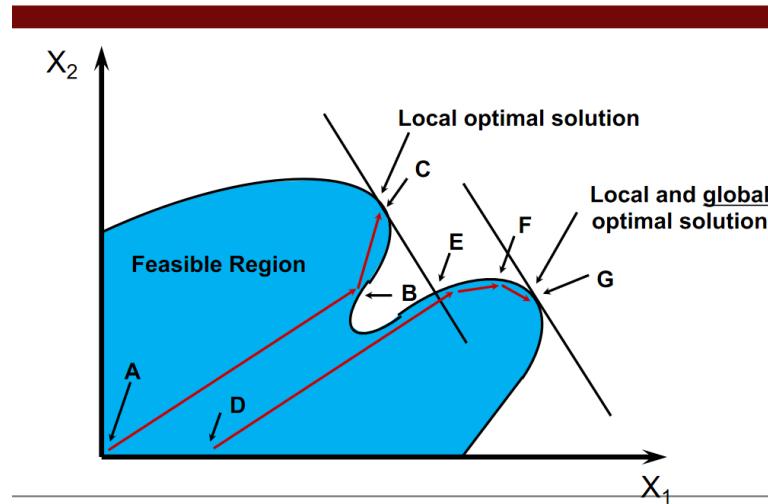


Figure 14.26: Local vs. Global Optimal Solution

Come faccio a distinguere una regione ammissibile convessa da una non convessa? In una regione ammissibile **convessa**, dati due punti qualsiasi li posso congiungere con un segmento e tutti i punti di quel segmento stanno all'interno di questa regione, come la prima. Questo non è vero nel caso la regione ammissibile sia **concava**: se prendo un punto sulla prima gobba e un punto sulla seconda e li unisco, non tutti i punti del segmento che li unisce appartengono alla regione ammissibile (ci sono degli avallamenti,

dei buchi). In questo caso posso partire dal punto A, determino la direzione più ripida che mi fa salire più velocemente, arrivo al punto B e dal punto B riparto verso il punto C trovando un punto di ottimo e fermandomi perché non riesco più a salire, non essendoci direzioni di miglioramento a partire dal punto C. Ma quello è un punto di ottimo **solo locale!** Dalla figura vediamo che il punto ottimo globale è rappresentato dal punto G: se parto dal punto A non ci arrivo con questo procedimento. Dovrei invece partire dal punto D. Abbiamo un problema: non basta più dire che si parte da un punto qualsiasi. Si può quindi rimanere bloccati in **ottimi locali**: è uno dei grossi problemi della NLP. Non ci sono algoritmi che mi permettono di superare questo problema (a meno che la funzione sia una funzione concava/convessa su un insieme concavo/convesso). In tutti gli altri casi, devo esplorare **diversi punti di partenza**: una possibilità è quella di partire da punti diversi e vedere se finisco sempre nello stesso punto o se finisco in punti diversi di ottimo locale, poi scelgo il migliore.

Quindi quando ho un problema di NLP dove la funzione non è convessa e quindi può avere **più ottimi locali** o quando devo ottimizzare su una regione ammissibile che non è convessa (avendo più ottimi locali), in qualche modo devo esplorare la mia regione per cercare di non venire intrappolato in ottimi locali e per riuscire a trovare la strada per arrivare all'ottimo globale.

FINE

DURANTE L'ESAME NON USEREMO R MA POTREMO USARE SOLO CARTA E PENNA E FARE FOTO AL COMPITO E INVIARLA QUINDI IL METODO DI RISOLUZIONE SARÀ PRINCIPALMENTE GRAFICO!